The University of Jordan

Authorization Form

I, **Mohammad A. Al-Quran**, authorize the University of Jordan to supply copies of my Thesis/ Dissertation to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature:

Date: 11 / 8 / 2011

الجامعة الأردنيـــــة
كلية الدراسات العليا

التاريـــخ: / /

نموذج رقم (18)
اقرار والتزام بالمعايير الأخلاقية والأمانة العلمية
وقوانين الجامعة الأردنية وأنظمتها وتعليماتها
لطلبة الماجستير

أنا الطالب: محمد عبدالله القرعان الرقم الجامعي: (8090429)

تخصص: الهندسة الميكانيكية الكليـــــــة: الهندسة والتكنولوجيا

عنوان الرسالة: ....

Dynamical Model Reduction Using Reducibility Matrix Optimization.

....

اعلن بأنني قد التزمت بقوانين الجامعة الأردنية وأنظمتها وتعليماتها وقراراتها السارية المفعول المتعلقة باعداد رسائل الماجستير عندما قمت شخصيا" باعداد رسالتي وذلك بما ينسجم مع الأمانة العلمية وكافة المعايير الأخلاقية المتعارف عليها في كتابة الرسائل العلمية. كما أنني أعلن بأن رسالتي هذه غير منقولة أو مستلة من رسائل أو كتب أو أبحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيسا" على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في الجامعة الأردنية بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

توقيع الطالب:.............................. التاريخ: ١١ / ٨ / ٢٠١١

DYNAMICAL MODEL REDUCTION USING REDUCIBILITY MATRIX OPTIMIZATION

By

Mohammad A. Al-Quran

Supervisor

Dr. Musa O. Abdalla

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Science in Mechanical Engineering

Faculty of Graduate Studies

The University of Jordan

August, 2011

COMMITTEE DECISION

This Thesis (**Dynamical Model Reduction Using Reducibility Matrix Optimization**) was Successfully Defended and Approved on 7/8/2011

Examination Committee	**Signature**
Dr. Musa O. Abdalla (Supervisor) Assoc. Prof. of Mechanical Engineering	
Dr. Mohamad N. Hamdan (Member) Prof. of Mechanical Engineering	
Dr. Mazen AlQeesi (Member) Prof. of Mechanical Engineering	
Dr. Nazzal S. Armouti (Member) Assoc. Prof. of Civil Engineering (University of Jordan)	

DEDICATION

To my great father

To my dear mother

To my brothers and sisters

ACKNOWLEDGMENT

First, I would like to express my sincere gratitude to my supervisor Dr. Musa Abdalla, for his advice and continuous encouragement throughout the period of this work.

Also, I would like to convey my appreciation to the members of the examination committee, for their time, and insight.

LIST OF CONTENTS

Committee Decision ……………………………………………………………..		ii
Dedication ……………………………………………………………………..		iii
Acknowledgment ……………………………………………………………..		iv
List of Contents ……………………………………………………………….		v
List of Figures ………………………………………………………………...		vi
List of Abbreviations ...……………………………………………………….		vii
List of Appendices …………………………………………………………….		ix
Abstract ……………………………………………………………………….		x
Chapter One	**Introduction …………………………………………….**	1
	1.1 Previous Work …………………………………..	2
	1.2 Thesis Layout ……………………………………	4
Chapter Two	**Mathematical Preliminaries ……………………………**	5
	2.1 State Space Representation ……………………...	5
	2.2 Steepest Descent Method ………………………..	7
	2.3 Reducibility Matrix Concept …………………….	9
	2.4 Hamiltonian Optimization with Equality Constraints …………………………………………..	10
	2.5 Kronecker Product ………………………………	11
	2.6 Singular Perturbation Method …………………..	12
	2.7 Model Reduction by Truncation Method ……….	13
Chapter Three	**Model Reduction……………………………………...**	15
	3.1 Model Reduction ………………………………..	15
	3.2 Reducibility matrix Based Reduction…..………..	17
	3.3 Hamiltonian Optimization..……………………...	20
	3.4 Optimization Using Steepest Descent Method ….	23
Chapter Four	**Model Reduction Application………………………...**	30
	4.1 Second Order System Application………………	30
	4.2 Special Third Order System……………………..	34
	4.3 Dynamical System with Some Complex Roots…	38
	4.4 Dynamical System with All Complex Roots……	40
	4.5 Fifth Order System Application…………………	42
Chapter Five	**Conclusions and Future Work……...…………………..**	47
References ……………………………………………………………………		48
Abstract (in Arabic) …………………………………………………………		68

LIST OF FIGURES

Figure 1: The effect of the eigenvalues on the response 6

Figure 2: Graphical description for Steepest Descent method 8

Figure 3: Model reduction concept 15

Figure 4: Reducibility matrix reduction techniques 20

Figure 5: Flow Chart for optimization technique Using LMI and Steepest Descent. 24

Figure 6: searching about the minimum starting from initial guess ($vecA_c$) and going in the direction of the unit gradient vector with step size equal α_k 29

Figure 7: Full 2nd order system response 31

Figure 8: Full and reduced (using steepest descent) system's response 32

Figure 9: Full and reduced system's response for small time range 33

Figure 10: Full and reduced order system response for 2nd order system 34

Figure 11: Third Order System Response 35

Figure 12: Full and reduced order systems 36

Figure 13: Full order, and reduced systems order using steepest and perturbation method 37

Figure 14: Full order system response for 3 by 3 system 38

Figure 15: Full and reduced system's response using steepest method 40

Figure 16: fourth order system with the reduction methods output 42

Figure 17: 5×5 full order system response 43

Figure 18: Full and reduced system's response using steepest method 44

Figure 19: Full order and reduced order (truncation method) for 5×5 system reduced to 3×3 46

LIST OF ABBREVIATIONS

A	System state matrix
$\tilde{A}$	Upper triangular matrix (*A* matrix after transformation)
A_c	A matrix that entries is chosen optimally
A_o	A matrix that contains omitted eigenvalues
A_r	A matrix that contains retained eigenvalues
$\breve{A}_r$	State matrix using truncation method
A_0	State matrix using singular perturbation method
B	Input matrix
$\tilde{B}$	*B* matrix after transformation
B_o	Omitted *B* matrix
B_r	Retained *B* matrix
$\breve{B}_r$	Input matrix using truncation method
B_0	Input matrix using singular perturbation method
C	System output matrix
$\tilde{C}$	*C* matrix after transformation
C_r	Retained *C* matrix
C_o	Omitted *C* matrix
$\breve{C}_r$	System output matrix using truncation method
C_0	System output matrix using singular perturbation method
D	System feed forward matrix
$\breve{D}_r$	System feed forward matrix using truncation method
D_0	System feed forward matrix using singular perturbation method
f	The constraint
f	The cost function
H	Hamiltonian function
P	Perturbation Matrix
u	System input
x	State vector
$\tilde{x}$	*x* states after transformation

$\tilde{x}_r$	Retained states
$\tilde{x}_o$	Omitted states
$\check{x}$	States using truncation method
$\bar{x}$	States using singular perturbation
y	Output vector
y_{full}	Full order system response
$y_{reduced}$	Reduced order system response
$\check{y}$	The response using truncation method
$\bar{y}$	The response using singular perturbation method
S	Searching direction for steepest descent method
S	Symmetric slack matrix
T	Transformation matrix for truncation
$vecA_c$	A_c as a column vector

Greek Symbols

α	Constant that updated every steepest descent loop
ε	A small positive number
λ	Lagrange multiplier

LIST OF APPENDICES

APPENDIX A: MATLAB CODE FOR REDUCING 2BY2 SYSTEM INTO 1BY1 USING STEEPESTDESCENT AND SINGULAR PERTURBATION 50

APPENDIX B: MATLAB CODE FOR REDUCING 3BY3 SYSTEM INTO 2BY2 USING STEEPEST DESCENT AND TRUNCATION 54

APPENDIX C: MATLAB CODE FOR REDUCING 3BY3 SYSTEM INTO 2BY2 USING STEEPEST DESCENT WITH COMPLEX EIGENVALUES 57

APPENDIX D: MATLAB CODE FOR REDUCING 4×4 SYSTEM INTO 2×2 USING STEEPEST DESCENT AND PERTURBATION METHOD WITH ALL COMPLEX EIGENVALUES 60

APPENDIX E: MATLAB CODE FOR REDUCING 5BY5 SYSTEM INTO 3BY3 USING STEEPEST DESCENT AND TRUNCATION 64

DYNAMICAL MODEL REDUCTION USING REDUCIBILITY MATRIX OPTIMIZATION

By
Mohammad A. Al-Quran

Supervisor
Dr. Musa O. Abdalla

ABSTRACT

A reducible matrix model reduction method is used to provide a systematic approach for systems model reduction. Selected eigenvalues may be preserved in the reduced order system; making the reduced system more physically accepted.

Steepest descent method was used to optimize a matrix coupling term that appears within the reducible matrix model reduction method. The proposed method was successfully verified and validated using computer simulations and it was compared with other model reduction techniques showing promising results.

Chapter One

Introduction

The new trend of today's structures is basically larger is better. Recently, many tall tower buildings are being built. A close look at Dubai and Qatar reveals such mega structures being erected. Unfortunately, as the structure complexity increases its corresponding control and health monitoring systems order increases as well.

Consequently, modeling of systems in real life usually results in high order systems especially for mechanical systems and structures. Sometimes it is needed to reduce the order of such systems because its complexity creates burdens during the analysis, simulation, and controller design stages. But many of the well developed reduction techniques make order reduction without paying attention to preserve the systems eigenvalues (natural frequencies). However, the eigenvalues are very important physical quantity because preserving them means preserving a global physical property of the system (Abdalla, 2009); which is critical for many engineering applications.

Hence, there is a great need for establishing model reduction techniques that have the ability to preserve such important physical quantities. In this work the reducibility matrix reduction novel technique is further developed to generate more physical meaningful reduced models. This well established technique uses the reducibility matrix method, which is an iterative procedure. In this work, an optimized or a closed form solution of the reducibility matrix reduction method is sought.

Creating a new optimization or closed form based technique for reducing dynamical systems order, provided preserving the system's natural frequencies (i.e. eigenvalues). Optimization techniques will be used to further enhance the well established reducibility

matrix reduction technique to eliminate the need to tune the coupling term (Abdalla and Alsmadi, 2006). The proposed technique will be compared with other well established model reduction procedure such as perturbation theory.

1.1 Previous Work

A close look at the literature reveals that there are many techniques for performing model reduction. Some of these techniques preserve the important eigenvalues and others don't preserve them, and all of them have some advantages and disadvantages. The first known model reduction method was proposed and performed by (Gyuan, 1965) and (Irons, 1965). They made a partition for mass and stiffness matrices into master and slave degrees of freedom (DOF), and had used a transformation matrix to reduce the dynamical model into only the master DOFs. That transformation was approximate because it neglected the inertia forces at slave DOFs (Cunedioglu, et al., 2005); hence the name was given as static condensation.

The model reduction methods can be classified into four groups (Papadopulos, 1996). One class depends on omitting selected DOFs such as Guyan and Irons as mentioned above. So the accuracy of this class is very sensitive to the selection of active DOFs. Both (Kammer, 1987) and (O'Callahan, et al.,1987) differently formulated the exact reduction method which includes the dynamical effects in contrast with Guyan. The method depends on a transformation matrix that has the full finite element modal and the active subset. Another method called modal method which is proposed initially by Davison (Davison, 1966). The importance of this method is the ability to deal with high order systems, so for the large order systems it is one of the very few applicable methods. In this method the fast modes

(i.e. eigenvalues too far away from the imaginary axis called fast or non-dominant) are neglected and the slow modes (eigenvalues near imaginary axis called slow or dominant) are preserved (Varga, 1994). But there are some disadvantages that reside in the computation of the modes, which may be costly if the analysis needs too many modes, and their results may be unacceptable due to the effects of truncated modes (Papadopulos, 1996). The third class is based on the Ritz vector methods, which was developed to eliminate the computation of the costly eigenvalue problems and to improve the accuracy, where eigenvectors are not the best choice (Edward, et al., 1986); so it considered as generalized approach replacing the eigenvectors by more generally defined Ritz vectors. The last fourth class is called component mode synthesis (CMS) based methods. CMS methods are a combination of the first aforementioned three methods (Papadopulos, 1996). It was first proposed by Hurty (1965), and it's suitable for modeling and simulation of large and complex systems (Sellgren, 2003). The basic approach of CMS is to divide the system into k subsystems and then using Ritz Vectors to establish a transformation matrix in order to reduce the order of the system.

The usage of reducibility matrix in model reduction is a novel technique, which was recently introduced (Abdalla and Alsmadi, 2006). The benefit of this method is that the system's eigenvalues are preserved; that is, the eigenvalues of the reduced order model are a subset of the eigenvalues of the full order model. This means that the reduced order model will have more physical reality because the preserving of the original eigenvalues means preserving the global physical properties of the system. In such systems that their eigenvalues represent the natural frequencies, such as mechanical systems and structures

and structures this becomes crucial (Abdalla, 2009). In this work, optimization techniques will be used to enhance the model reduction produced from reducibility matrix technique.

1.2 Thesis Layout

Chapter1 highlights about the importance of the reduction process in general and the importance of preserving the eigenvalues for the mechanical system, also it describes some benefits for the novel techniques used here. Some of previous studies is mentioned here and the different between our study and that studies is discussed. Mathematical preliminaries are provided in chapter two that will give us the basics for solving the problem. Chapter three explains how to reduce the original model using a numerical technique (Steepest Descent Method).

In chapter four different numerical examples are solved and they are compared with the original systems response and with other techniques (the singular perturbation method and the truncation method). Finally, chapter five conclude the work done with conclusions.

Chapter Two

Mathematical Preliminaries

2.1 State Space Representation

Modern control theory is based on the description of dynamical systems equations in terms of *n* first order equations; which unify the mathematical representation of these system equations.

Hence, increasing in number of state variables, number of inputs, or number of outputs for the dynamical system does not increase the complexity of the state equations. This representation may be given in the following form

$$\begin{aligned} \dot{x} &= A_{n\times n}x_{n\times 1} + B_{n\times m}u_{m\times 1} \\ y &= C_{p\times n}x_{n\times 1} + D_{p\times m}u_{m\times 1} \end{aligned} \tag{1}$$

Where $x \in \mathcal{R}^{n\times 1}$is the state vector, $y\in \mathcal{R}^{p\times n}$ is the output vector, $u\in \mathcal{R}^{m\times 1}$ is the input, $A \in \mathcal{R}^{n\times n}$is the state matrix, $B\in \mathcal{R}^{n\times m}$ is the actuator matrix, $C \in \mathcal{R}^{p\times n}$is the sensor matrix, and $D \in \mathcal{R}^{p\times m}$is the feedback matrix (Ogata,2002).

In this approach the dynamical system behavior is stored in the system matrix *A,* while the *B* matrix represents where in the system the actuators are placed. On the other hand the output equation has the *C* matrix, which shows how are the sensors are placed in the dynamical system and which states (physical parameters) is measuring (i.e. displacement, velocity, or acceleration for mechanical systems as an example). Therefore, all the dynamical behavior of the system is captured and fully encoded by Equation 1.

On the other hand, it turns out that the stability of the dynamical system may be determined by finding the eigenvalues of the A matrix, where negative real eigenvalues $\sigma(A) < 0$ means stable system.

Note that, the location of the eigenvalues in the s-plane dictates the nature and speed of the system response. That is, far eigenvalues (poles) from the origin reflects fast modes while close ones represents slower modes. Figure 1 depicts an example of three real eigenvalues for a first order system

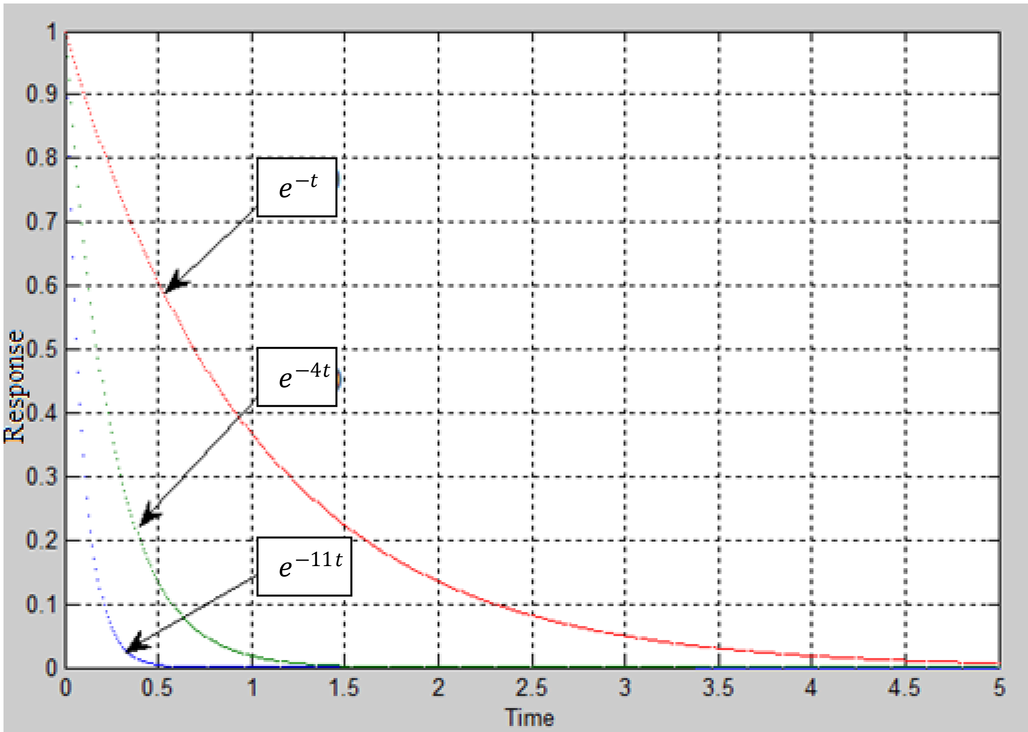


Figure 1: The effect of the eigenvalues on the response

It is noted that the e^{-11t} decreasing very rapidly and the effect reaches zero at about 0.5s while e^{-4t} has longer effect than e^{-11t}, and e^{-1t} has the largest effect. This concept of fast modes has the benefit for choosing the eigenvalues that have the most effect to preserve it during the reduction.

The solution for the State Space Representation is given in the following Equation (Ogata, 2002)

$$x(t) = e^{At}x(0) + \int_{0}^{t} e^{A(t-\tau)}Bu(\tau).\,d\tau \qquad (2)$$

Where $e^{At} \in \mathcal{R}^{n\times n}$ is the instantaneous exponential matrix (i.e. transient matrix) $\emptyset(t) = e^{At}$ and the first term represents response due to initial conditions (i.e. transient) while the integral term represent the forced response (i.e. steady state), and $x(0)$ is the initial conditions.

Or the response can be found numerically using Matlab function while accounting to the kind of the input the system is subjected to.

$$sys = ss(A, B, C, D)$$

$$[y, t, x] = step(sys, t) \quad \text{for step input}$$

$$[y, t, x] = initial\,(sys, t) \quad \text{for initial conditions} \qquad (3)$$

$$[y, t, x] = impulse\,(sys, t) \quad \text{for impulse input ... etc}$$

From these commands the x values may be estimated at every time frame.

2.2 Steepest Descent Method

There are many techniques that are used for searching the solution space to optimize a certain cost function. One of the efficient searching techniques is the steepest descent.

It is basically a method that is used for finding a function local minimum value starting from an initial guess x_o. Figure 2 depicts a graphical illustration of the steepest descent method

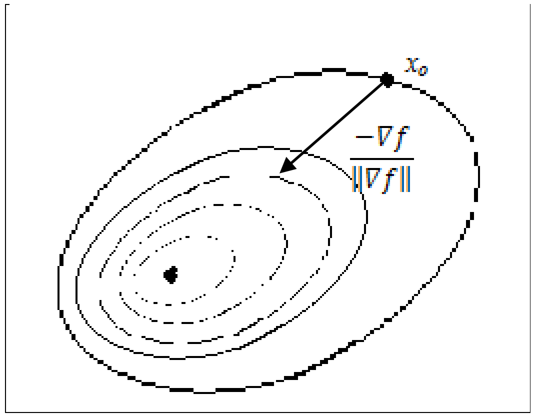


Figure 2: Graphical description for Steepest descent method

Starting from initial guess the gradient direction (unit vector) $\frac{-\nabla f}{\|\nabla f\|}$ is found to determine the direction of the minimum. A summary of the basic for steepest descent algorithm is as follows

1. Make an initial guess x_o.
2. Compute the search direction.

$$S_k = \frac{-\nabla f(x_k)}{\|\nabla f(x_k)\|} \tag{4}$$

3. Perform a single parameter minimization of the function $\emptyset(\alpha) = f(x_k + \alpha S_k)$ on the interval *[0,b]*, this will give $\alpha = h_{min}$, where a local minimum for $\emptyset(\alpha)$ occurs.
4. Construct the next point $x_{k+1} = x_k + h_{min} S_k$.
5. Check the convergence $\|x_{k+1} - x_k\| < \epsilon$.
6. Go to 2.

Note that the norm of a vector which is a function of time *g (t)* is given by

$$\|g(t)\| = \sqrt{\int_0^{tf} g(t)^T g(t).\, dt}$$

2.3 Reducibility Matrix Concept

Definition (Anton, 2005): a matrix $A \in M_n$ is called reducible if either

a. $n=1$ and $A=0$; or

b. $n \geq 2$, there is a permutation matrix $P \in M_n$, and there is some integer r with $1 \leq r \leq n-1$ such that

$$P^T AP = \begin{bmatrix} U & V \\ 0 & W \end{bmatrix} \tag{5}$$

Where $U \in M_r,\ V \in M_{n-r}\ ,\ W \in M_{r,n-r}\ ,\ and\ 0 \in M_{n-r}\ is\ the\ zero\ matrix.$

This transformation is very important for the work this thesis, but it was modified to another form to be more useful for the reduction process. It was used as a similar transformation$f: R^{n\times n} \rightarrow R^{n\times n}$, where f is a linear operator, which is defined by $f(A) = P^{-1}AP$, where the permutation matrix P is selected using an optimization technique called Linear Matrix Inequalities (LMI).

The LMI optimization problem is casted as follows

$$\begin{gathered} min_P \|P - P_o\| \\ Subjected\ to\ \left\|P^{-1}AP - \tilde{A}\right\|\ < \varepsilon \end{gathered} \tag{6}$$

Which is written in LMI equivalent form as

$$min_S \, trace(S)$$

$$Subjected\ to \begin{bmatrix} S & P-P_o \\ (P-P_o)^T & I \end{bmatrix} > 0 \tag{7}$$

$$\begin{bmatrix} \varepsilon_1^2 I & P^{-1}AP-\tilde{A} \\ \left(P^{-1}AP-\tilde{A}\right)^T & I \end{bmatrix} > 0$$

Where S is a symmetric slack matrix. Matlab was used to implement this LMI base transformation.

2.4 Hamiltonian Optimization with Equality Constraints

Assume the scalar performance index *L(x,u)*, where *x* is the states and *u* is the vector that is to be optimized. So the problem is to choose a vector *u* such that the value of *L(x,u)* is minimized and at the same time satisfies the constraint equation (Lewis and Syrmos, 1995)

$$f(x,u) = 0 \tag{8}$$

After some derivations the Hamiltonian equation has the following final form

$$H(x,u,t) = L(x,u,t) + \lambda^T f(x,u,t) \tag{9}$$

Where λ is the Lagrange multiplier vector.

To find the optimal solution for *u* one need to solve the following set of DEs simultaneously

States equation

$$\dot{x} = \frac{\partial H}{\partial \lambda} = f \tag{10}$$

Costate equation

$$-\dot{\lambda} = \frac{\partial H}{\partial x} = \frac{\partial f^T}{\partial x}\lambda + \frac{\partial L}{\partial x} \tag{11}$$

Stationary conditions

$$0 = \frac{\partial H}{\partial u} = \frac{\partial f^T}{\partial u}\lambda + \frac{\partial L}{\partial u} \tag{12}$$

2.5 Kronecker Product

Consider a matrix $M \in \mathcal{R}^{m\times n}$ =*[m_{ij}]* and a Matrix $N \in \mathcal{R}^{r\times s}$ =*[=[n_{ij}]*. The Kronecker product of the two matrices is defined as the partitioned matrix (Graham, 1981)

$$M \otimes N = \begin{bmatrix} m_{11}N & m_{12}N & \dots & m_{1n}N \\ m_{21}N & m_{21}N & \dots & m_{2n}N \\ \vdots & \vdots & \ddots & \vdots \\ m_{m1}N & m_{m2}N & \dots & m_{mn}N \end{bmatrix} \tag{13}$$

$M \otimes N$ is seen to be ($mr \times ns$). The main disadvantage of the Kronecker product is the dimensionality.

On the other hand the Kronecker sum for the same matrices is defined as

$$M \oplus N = M \otimes I_m + I_n \otimes N \tag{14}$$

Where the $I_m \in \mathcal{R}^{m\times m}$ *and* $I_n \in \mathcal{R}^{n\times n}$ are the identities matrices.

It turns out that the Kronecker matrix concepts enable us to solve complicated matrix equations. A good example is the following Lyapunov like matrix algebraic equation.

$$MX + XN = G \tag{15}$$

The solution using the Kronecker concepts may be given as (Graham, 1981)

$$(N^T \oplus M)vecX = vecG \tag{16}$$

Where $vecX$ contains the X components in a column vector form (as stacked vector).

Equation will be appeared later in this thesis and the solution provided by Equation 16 will be handy.

2.6 Singular Perturbation Method

Singular perturbation method is one of the methods that is used to reduce dynamical system orders. Starting with a dynamical system that is represented using state space as in Equation 1, the perturbation method provides an equivalent system of the form

$$\begin{aligned} \dot{x}_1 &= A_{11}x_1(t) + A_{12}x_2(t) + B_1u(t) \\ \varepsilon\dot{x}_2 &= A_{21}x_1(t) + A_{22}x_2(t) + B_2u(t) \\ y(t) &= C_1x_1(t) + C_2x_2(t) \end{aligned} \tag{17}$$

Where ε is a small positive scalar (Abdalla and Alsmadi, 2006) and A_{ij} are partitioned parts of the A Matrix. Now, letting $\varepsilon = 0$ means neglecting the fast modes in the system (i.e. eigenvalues with most negative real parts), this will result in the following model

$$\begin{aligned} \dot{\bar{x}}_1 &= A_0\bar{x}_1(t) + B_0u(t) \\ \bar{y}(t) &= C_0\bar{x}_1(t) + D_0u(t) \end{aligned} \tag{18}$$

Where the new matrices are

$$A_0 = A_{11} - A_{12}A_{22}^{-1}A_{21}$$
$$B_0 = B_1 - A_{12}A_{22}^{-1}B_2 \tag{19}$$
$$C_0 = C_1 - C_2A_{22}^{-1}A_{21}$$
$$D_0 = -C_2A_{22}^{-1}B_2$$

Note that the A_{22} Matrix must be invertible.

That represents the reduced dynamical system after dropping the fast modes. These fast modes have minimal contribution to the system dynamics as time passes by.

2.7 Model Reduction by Truncation Method

Given a linear system in Equation 1 and consider the following transformation (Skelton, Oliveira, and Han)

$$[T_1 \quad T_2]\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = x\,,\, and\ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} R_1^T \\ R_2^T \end{bmatrix} x\,,\, then\ T_1R_1^T + T_2R_2^T = I \tag{20}$$

This transformation will convert the original system into

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} \breve{A}_{11} & \breve{A}_{12} \\ \breve{A}_{21} & \breve{A}_{22} \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} \breve{B}_1 \\ \breve{B}_2 \end{bmatrix} u$$
$$y = [\breve{C}_1 \quad \breve{C}_2] + Du \tag{21}$$

Where

$$\begin{bmatrix} \breve{A}_{11} & \breve{A}_{12} \\ \breve{A}_{21} & \breve{A}_{22} \end{bmatrix} = \begin{bmatrix} R_1^TAT_1 & R_1^TAT_2 \\ R_2^TAT_1 & R_2^TAT_2 \end{bmatrix}, \qquad \begin{bmatrix} \breve{B}_1 \\ \breve{B}_2 \end{bmatrix} = \begin{bmatrix} R_1^TB \\ R_2^TB \end{bmatrix}, \qquad [\breve{C}_1 \quad \breve{C}_2] = [CT_1 \quad CT_2] \tag{22}$$

It is wanted to preserve the state z_1 only; so the following represent the state-space after reduction

$$\left(\breve{A}_r, \breve{B}_r, \breve{C}_r, \breve{D}_r\right) = (\breve{A}_{11,} \breve{B}_1, \breve{C}_1, \breve{D}) = (R_1^T A T_1, R_1^T B, C T_1, D) \tag{23}$$

By constructing $T_1 R_1^T = I$ and assuming that J is a square and nonsingular matrix then

$$T_1 = J^T R_1 (R_1^T J^T R_1)^{-1} \tag{24}$$

Chapter Three

Model Reduction

3.1 Model Reduction

The reduction of the systems order with preserving its dynamical characteristics is becoming more and more essential especially for nowadays Mega structures. This reduction on systems order is not only useful for performing system's analysis but also for building useful controllers.

The main objective of all reduction techniques is to preserve the original system dynamics. That is the dynamical behavior of the reduced system should resemble the full order system for a certain working window. Typically, systems tend to be used in a working window; hence not all the system's dynamics is excited. For example upper modes need a lot of energy to excite.

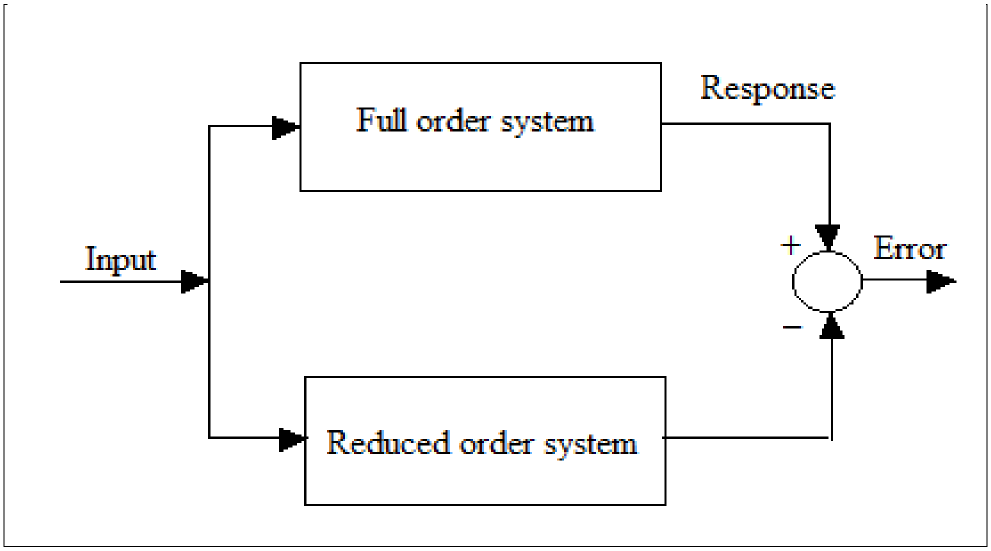


Figure 3: Model reduction concept

Figure 3 illustrates the reduction conceptual methodology. The reduced system needs to capture as much dynamics of the original system as possible. Even though there exists many techniques that perform the reduction nicely, but it destroys the physical meaning of the system. A good example for that is reducing a structure model. Now, if the natural frequencies set of the reduced and the full are not intersecting then we are having a big problem!

On the other hand, the deformation of a structure takes at a given natural frequency (mode shape) poses another constraint for the reduction process. It turns out that not only the matching of the dynamics (response) of the reduced and full systems is needed, but also one needs to enforce the frequencies (eigenvalues) matching and mode shapes (eigenvectors) matching for a structure.

In this work, the low eigenvalues or the structure fundamental frequencies matching is enforced.

Hence, the reduction problem boils down to the following optimization problem

$$
\begin{aligned}
& min_P \|y - y_r\| \\
\textit{Subjected to} \quad & \dot{x} = A_{n\times n}x(t) + Bu(t) \\
& \dot{x}_r = A_{r_{m\times m}}x_r(t) + B_r u(t) \\
& y = Cx + Du \\
& y_r = C_r x_r + D_r u \\
& \sigma(A_r) \subset \sigma(A)
\end{aligned}
\tag{25}
$$

Where the $x \in \mathcal{R}^n$ and $x_r \in \mathcal{R}^m$ are the states vectors for full and reduced systems, respectively. While, y and y_r are the systems output for full and reduced systems,

respectively. Finally, the eigenvalues of the reduced system should be a subset of the full system, $\sigma(A_r) \subset \sigma(A)$.

3.2 Reducibility Matrix Based Reduction

The reducibility matrix reduction technique was established to preserve system's selected eigenvalues within the reduced system (Abdalla and Alsmadi, 2006). An overview of the technique is provided here for reference.

Consider a state space representation for a dynamical system is given by

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \quad (26)$$

Where *x(t)* is the state vector, *u(t)* is the control input vector, *y(t)* is the measured output vector, and *A*, *B*, *C*, and *D* are real constant matrices.

Using matrix reducibility concept (see mathematical preliminaries) the system matrix *A* can be converted into upper triangular matrix $\tilde{A}$ using Linear Matrix Inequalities (LMI), that is a transformation of the form $A = P\tilde{A}P^{-1}$ is utilized to accomplished that (Abdalla and Alsmadi, 2006). Where the transformation *P* matrix is computed using LMI. Applying this transformation into the State Space Representation, one gets

$$\dot{x} = P\tilde{A}P^{-1}x + Bu$$
$$y = Cx + Du \quad (27)$$

Where the new transformation system matrix is given as $\tilde{A} = \begin{bmatrix} A_r & A_C \\ 0 & A_o \end{bmatrix}$

By multiplying by P^{-1} one gets

$$P^{-1}\dot{x} = \tilde{A}P^{-1}x + P^{-1}B \tag{28}$$

Now assuming that $\tilde{x}(t) = P^{-1}x,\ and\ \tilde{B} = P^{-1}B$, then the transformed model can be written as

$$\dot{\tilde{x}} = \tilde{A}\,\tilde{x} + \tilde{B}u$$
$$y = \tilde{C}\tilde{x} + Du \tag{29}$$

Where $\tilde{C} = CP$ and $\tilde{x} = \begin{bmatrix}\tilde{x}_r \\ \tilde{x}_o\end{bmatrix}$ is the transformed state vector, $\tilde{x}_r$ is the retained states while $\tilde{x}_o$ are the omitted states.

Hence, the partitioned system can be written in the following form

$$\begin{bmatrix}\dot{\tilde{x}}_r(t) \\ \dot{\tilde{x}}_o(t)\end{bmatrix} = \begin{bmatrix}A_r & A_c \\ 0 & A_o\end{bmatrix}\begin{bmatrix}\tilde{x}_r(t) \\ \tilde{x}_o(t)\end{bmatrix} + \begin{bmatrix}B_r \\ B_o\end{bmatrix}u(t)$$
$$\tilde{y}_r(t) = \begin{bmatrix}C_r & C_o\end{bmatrix}\begin{bmatrix}\tilde{x}_r(t) \\ \tilde{x}_o(t)\end{bmatrix} + \begin{bmatrix}D_r \\ D_o\end{bmatrix}u(t) \tag{30}$$

Where,

r Indicates the retained eigenvalues, o indicates the omitted eigenvalues, A_r matrix contains retained eigenvalues, A_o matrix contains omitted eigenvalues, A_c is a coupling matrix that needed to be selected optimally, $\tilde{x}_r$ are the retained states, and $\tilde{x}_o$ are the omitted states.

Expanding Equation 30 yields the following coupled DEs:

$$\dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + A_c\tilde{x}_o(t) + B_r u(t)$$
$$\dot{\tilde{x}}_o(t) = A_o\tilde{x}_o(t) + B_o u(t) \tag{31}$$

Using static condensation that setting $\dot{x}_o(t) = 0$ one gets

$$\tilde{x}_o(t) = -A_o^{-1}B_o u(t) \quad \text{(a)}$$

$$\dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + [-A_c A_o^{-1}B_o + B_r]u(t) \quad \text{(b)} \tag{32}$$

Now, there will be some deviations between the models response (dynamics) due to the static condensation. To overcome this problem the coupling matrix A_c needs to be tuned. An optimization procedure is needed to optimize the selection of the entries of the A_c matrix. The optimization problem may be casted as follows

$$min_{A_c}\|y_{full} - y_{reduced}\|$$
$$Subjected\ to\ \dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + [-A_c A_o^{-1}B_o + B_r]u(t) \tag{33}$$

Where,

$y_{full} = Cx$, is the full order response.

$y_{reduced} = C_r\tilde{x}_r - C_o A_o^{-1}B_o u(t)$, is the reduced system response.

Figure 4 depicts a block diagram summarizing the Equation 32

Proper selection of the A_c matrix is critical for the reduction process to give good tracking of the dynamics of the original full order system.

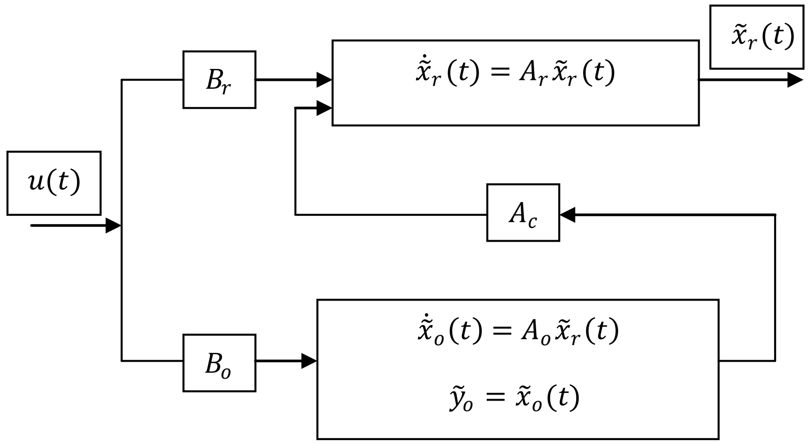


Figure 4: Reducibility matrix reduction techniques

Finally, the partition $\tilde{A} = \begin{bmatrix} A_r & A_c \\ 0 & A_o \end{bmatrix}$ matrix has the A_r matrix which has the selected user eigenvalues (retained) while A_o the omitted eigenvalues.

In the original work of the reducibility matrix the authors did not provide a method to select the entries of the coupling matrix A_c. In this work, two techniques are used to provide optimized A_c matrix in order to enhance the reduction process. The first technique is based on Hamiltonian optimization, which seeks a closed form solution for the optimization problem, while the second one utilized numerical techniques based on the steepest descent method.

3.3 Hamiltonian Optimization

The optimization problem in Equation 33 can be written in the following form (Lewis and Syrmos, 1995)

$$\min_{A_c} J = \frac{1}{2}\int_0^{t_f} (Cx - C_r\tilde{x}_r + C_o A_o^{-1} B_o u(t))^2 \, dt$$
$$\textit{Subjected to} \quad \dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + [-A_c A_o^{-1} B_o + B_r]u(t) \tag{34}$$

Where J is the performance index function (i.e. cost function), which has the integral of the square of the error to be minimized over A_c subject to the previously derived reducible matrix equation constraint.

Implementing the Hamiltonian Optimization on gets:

$$H = \frac{1}{2}\left(Cx - C_r\tilde{x}_r + C_o A_o^{-1} B_o u(t)\right)^2 + \lambda^T\left(A_r\tilde{x}_r + (B_r - A_c A_o^{-1} B_o)u(t)\right) \qquad (35)$$

Where λ is the Lagrange multiplier with size $(v \times 1)$, where v is the number of retained eigenvalues, to perform the optimization the Hamiltonian equation must be differentiate with respect to $\tilde{x}_r,\ \lambda, and\ A_c$

$$-\dot{\lambda} = \frac{\partial H}{\partial \tilde{x}_r} \Rightarrow -\dot{\lambda} = A_r^T\lambda + 2C_r^T\left(Cx - C_r\tilde{x}_r + C_o A_o^{-1} B_o u(t)\right) \qquad \text{(a)}$$

$$\dot{\tilde{x}}_r = \frac{\partial H}{\partial \lambda} \Rightarrow \dot{\tilde{x}}_r = A_r\tilde{x}_r + (B_r - A_c A_o^{-1} B_o)u(t)\ , The\ Constraint \qquad \text{(b)} \qquad (36)$$

$$0 = \frac{\partial H}{\partial A_c} \qquad \text{(c)}$$

Equation (*36-c*) can't be found directly because from section 3.2 it is noted that $\tilde{x},\ \tilde{B},\ \tilde{C},$ $and\ \tilde{A}$ are dependent on the permutation matrix, and the P matrix depends on the A_c matrix according to the transformation matrix

$$A = P\tilde{A}P^{-1} \Rightarrow A = P\begin{bmatrix} A_r & A_C \\ 0 & A_o \end{bmatrix}P^{-1} \qquad (37)$$

So that each of these parameters $(\tilde{x}_r, \tilde{x}_o, \tilde{B}_r, \tilde{B}_o, \tilde{C}_r,\ and \tilde{C}_o)$ depends on the A_c matrix. The Hamiltonian Equation 35 after substitution becomes:

$$H = \frac{1}{2}\left(Cx - CPK_1^T\tilde{x}_r + CPK_2^T A_o^{-1} K_2 P^{-1} Bu(t)\right)^2 + \lambda^T\left(A_r\tilde{x}_r + (K_1P^{-1}B - A_cA_o^{-1}K_2P^{-1}B)u(t)\right) \quad (38)$$

Where $k_1 = [I_{v\times v} \quad 0_{v\times w}]$, such that *v* is number of retained states, and *w* is the number of omitted states, and $k_2 = [0_{w\times w} \quad I_{w\times v}]$

But from Equation 37 it is noted that it's not possible to separate P *from* A_c and assuming that $P = some\ form\ of\ A_c$ does not lead to any improvement.

Many methods are tested to separate P and A_c such as Kronecker Product (Graham, 1981) as follows:

Equation 37 can be simplified to $P^{-1}(-A) + \tilde{A}P^{-1} = 0$ (sec. 2.5) then

$$\left((-A)^T \otimes I_n + I_n \otimes \tilde{A}\right) vec\ P^{-1} = 0$$

$$\text{then} \left((-A)^T \otimes I_n + \begin{bmatrix} \tilde{A} & 0 & 0 & 0 \\ 0 & \tilde{A} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{A} \end{bmatrix}\right) vec\ P^{-1} = 0 \quad (39)$$

Or Equation 37 can be simplified to $(-A)P + P\tilde{A} = 0$ then

$$\left(\tilde{A}^T \otimes I_n + I_n \otimes (-A)\right) vec\ P = 0 \quad (40)$$

But In the two cases (39 and 40) it is noted that the separation is very tough and complicated also as in Equations 39 and 40 the *P* matrix is defined as a column vector which increases the complexity of that problem. So that an alternative method is used to solve this problem numerically using steepest descent for following two reasons:

1. The toughness of making the *P* matrix equal some form of A_c matrix.

2. The P matrix was found from the previous research (Abdalla and Alsmadi, 2006) numerically using Linear Matrix Inequalities (LMI).

Finally, it is believed that this method needs further investigation to reach a closed form solution. Future work on this matter could be a good PhD dissertation especially using calculus of variation.

3.4 Optimization Using Steepest Descent

In the previous section we have shown that a closed form solution was not possible without complex mathematics using calculus of variation. In this part we are showing a numerical based optimal solution using steepest descent.

Figure 5 illustrates the iterative algorithm that utilizes steepest descent to provide an optimal coupling matrix A_c. the algorithm starting with random initial guess for the A_c matrix.

Then the transformation matrix P is computed using Linear Matrix Inequalities (LMI). A step input response for both systems; full and reduced is computed using Matlab and the corresponding outputs of the systems are compared. In the case of agreement then optimal A_c matrix is reached. On the other hand, if disagreement of the outputs (compared through Euclidean norm) then further search of the solution space is needed.

In this work steepest descent is utilized to provide the proper guidance through the search quest. Steepest descent provides step size and best direction in the solution space towards a minimal cost function. The direction is provided through the gradient of the cost function while the step size is computed as α_k.

To determine α_k (step size) consider $h(\alpha_k) = f(vecA_{C_k} - \alpha_k \frac{\nabla f_k}{\|\nabla f_k\|})$ where $vecA_c$ means the matrix A_c in a column vector form, so the optimum α_k is the value at which $h(\alpha_k)$ *is minimized*, *i.e.* $dh(\alpha_k)/d\alpha_k = 0$, where f is the cost function (that wanted to be minimized), and ∇f_k means the gradient of the cost function at every iteration with respect to $vecA_c$ elements.

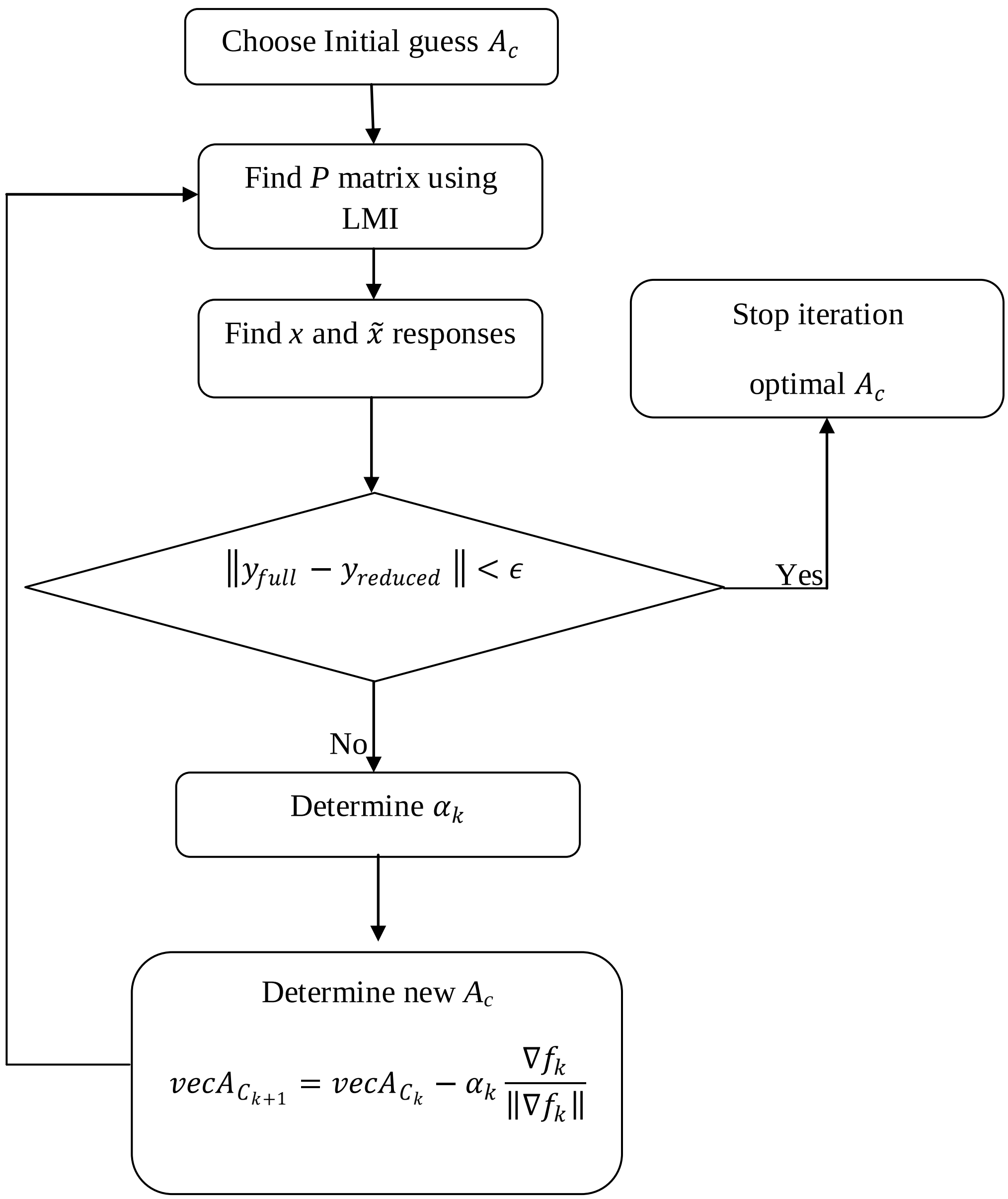


Figure 5: Flow Chart for optimization technique Using LMI and steepest descent.

From Figure 5 it is noted that the x *and* $\tilde{x}_r$ are needed. The two responses may be calculated numerically or analytically by solving Equation 41 (the state space representation for the full order system) and Equation 42 (the state space representation for the reduced order system) respectively.

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \tag{41}$$

Where *x(t)* is the state vector, *u(t)* is the control input vector, *y(t)* is the measured output vector, and *A, B, C,* and *D* are real constant matrices.

$$\dot{\tilde{x}}_r(t) = A_r\tilde{x}_r(t) + [-A_cA_o^{-1}B_o + B_r]u(t)$$
$$y_r = C_r\tilde{x}_r - C_oA_o^{-1}B_ou(t) \tag{42}$$

Where y_r is the measured output vector for the reduced order system, $\tilde{x}_r$ the state vector for the reduced system, and $A_r, A_o, A_c, B_r, B_o, C_r,$ *and* C_o are discussed in Sec. 3.2.

The two cases (numerically or analytically) are discussed as follows:

1. **Numerical solution for x *and* $\tilde{x}_r$**

   The numerical solution for x *and* $\tilde{x}_r$ can be found using Matlab order depends on the kind of the input *u(t)*, in our case the order s*tep* for step input is used as a testing signal, then the solution of *x* from Equation 40 becomes as in Equation 43

$$sys = ss(A, B, C, D)$$
$$[y, t, x] = step(sys, t_f) \tag{43}$$

   Where *ss* indicates state space, and t_f is the final time. Equation 43 will solve the y, *and* x at every section of time.

And the solution of $\tilde{x}_r$ from Equation 42 becomes

$$sys = ss(A_r, B_r - C_o A_o^{-1} B_o, C_r, -C_o A_o^{-1} B_o)$$
$$[y_r, t, \tilde{x}_r] = step(sys, t_f) \quad (44)$$

Where *ss* indicates state space, and t_f is the final time.

2. Analytical solution for *x and* $\tilde{x}_r$

Note that from steepest descent method the gradient of the cost function with respect to A_c elements is needed, so to find the derivative of the cost function *f* it is needed to find *f* analytically, but *f* ,as in Equation 45, depends on x *and* $\tilde{x}_r$; so, it is necessary to find x *and* $\tilde{x}_r$ analytically.

$$f = \|Cx - C_r\tilde{x}_r + A_c A_o^{-1} B_o\| = \sqrt{\int_0^{t_f} (Cx - C_r\tilde{x}_r + A_c A_o^{-1} B_o)^2 \, dt} \quad (45)$$

To find the values of x *and* $\tilde{x}_r$ Equations 41 and 42 have to be solved analytically as follows:

Assume that the input *u* is step input (as a testing signal) then *x* from solving Equation 40 one gets

$$homogenus\ solution:\ x_h = e^{At} L_1 \text{ where } L_1 \text{ is constant}$$
$$Particular\ solution: let\ \ x_P = L_2 \text{ where } L_2 \text{ is constant} \quad (46)$$

Then $L_1 = x(0) - A^{-1}B, and\ L_2 = -A^{-1}B$. The general solution is the sum of the homogenous and particular solutions as in Equation 47

$$x = e^{At}(x(0) + A^{-1}B) - A^{-1}B \quad (47)$$

The value of $\tilde{x}_r$ can be found from Equation 42 in the same procedure

$$\tilde{x}_r = e^{A_r t}(\tilde{x}_r(0) + A_r^{-1}(B_r - A_C A_o^{-1} B_o)) - A_r^{-1}(B_r - A_C A_o^{-1} B_o) \tag{48}$$

Note that the matrices A_r, *and* A_o are constant and selected at the beginning, where A_r contains the retained eigenvalues and A_o contains the omitted eigenvalues, and A_c is selected optimally using steepest descent method.

And all of these matrices ($\tilde{x}_r, B_r, B_o, C_r,$ *and* C_o) are function of the transformation matrix P as in sec. 3.2. So if one assume $k_1 = [I_{v\times v} \quad 0_{v\times w}]$, and $k_2 = [0_{w\times w} \quad I_{w\times v}]$ where the subscript v is number of retained states, and subscript w is the number of omitted states, then these matrices can be given in the following form

$$\begin{gathered} \tilde{x}_r(0) = [I_{m\times m} \quad 0_{m\times l}]\tilde{x}(0) = k_1 P^{-1} x(0) \\ B_r = k_1 \tilde{B} = k_1 P^{-1} B \\ B_o = k_2 \tilde{B} = k_2 P^{-1} B \\ C_r = \tilde{C} k_1^T = C P k_1^T \\ C_o = \tilde{C} k_2^T = C P k_2^T \end{gathered} \tag{49}$$

Where $\tilde{B} = \begin{bmatrix} B_r \\ B_o \end{bmatrix}$, $\tilde{x} = \begin{bmatrix} \tilde{x}_r \\ \tilde{x}_o \end{bmatrix}$, and $\tilde{C} = [C_r \quad C_o]$

Equations in 49 is substituted in Equation 48 and then Equation 48 and Equation 47 is substituted in Equation 45 to get Equation 50

$$\begin{aligned} f = \Bigg(\int_0^{t_f} \{ & C(e^{At}(x(0) + A^{-1}B) - A^{-1}B) \\ & - CPk_1^T(e^{A_r t}(k_1 P^{-1} x(0) + A_r^{-1}(k_1 P^{-1} B - A_C A_o^{-1} k_2 P^{-1} B)) \\ & - A_r^{-1}(k_1 P^{-1} B - A_C A_o^{-1} k_2 P^{-1} B)) + A_c A_o^{-1} k_2 P^{-1} B \}^2 \, dt \Bigg)^{0.5} \end{aligned} \tag{50}$$

As mentioned above in Figure 5 at the beginning there is an initial guess for A_c matrix then the value of the transformation matrix P is found using LMI, then the responses x *and* $\tilde{x}_r$ is found using the numerical method (Equations 43 and 44) to calculate $\|y_{full} - y_{reduced}\|$. To find a new value for A_c using steepest descent it is needed to find the gradient for the cost function by differentiate Equation 50 (using the analytical solution for x *and* $\tilde{x}_r$) with every component of the A_c as follows

$$A_c = \begin{bmatrix} a_{c11} & \cdots & a_{c1v} \\ \vdots & \ddots & \vdots \\ a_{cv1} & \cdots & a_{cvv} \end{bmatrix} \tag{51}$$

Where v indicates the retained eigenvalues.

Note that the cost function f in Equation 50 for SISO system (single input single output system) is scalar (1 × 1 system), that means f is a function of the components of the $vecA_c$ matrix; so it is easy to find the gradient of f with respect to the $vecA_c$ matrix components.

The $vecA_c$ is defined as

$$vecA_c = \begin{bmatrix} a_{c11} \\ a_{c21} \\ \vdots \\ a_{cv1} \\ \vdots \\ a_{c1v} \\ \vdots \\ a_{cvv} \end{bmatrix} \tag{52}$$

Then the gradient of the cost function f is

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial a_{c11}} \\ \frac{\partial f}{\partial a_{c21}} \\ \vdots \\ \frac{\partial f}{\partial a_{cv1}} \\ \vdots \\ \frac{\partial f}{\partial a_{c1v}} \\ \vdots \\ \frac{\partial f}{\partial a_{cvv}} \end{bmatrix} \tag{53}$$

Then the norm of this gradient is found to determine a unit directional vector that directed towards the local minimum for the cost function from the initial guess for $vecA_c$, Figure 6 shows the initial guess and how the unit gradient vector is directed towards the solution to be as close as possible to the minimum point.

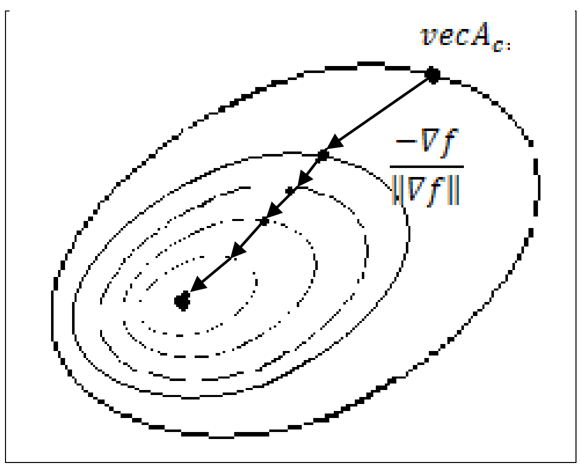


Figure 6: searching about the minimum starting from initial guess ($vecA_c$) and going in the direction of the unit gradient vector with step size equal α_k

Numerical examples are needed to check the validity of this method and to compare it with other will known methods.

Chapter Four

Model Reduction Application

The optimized reducible matrix reduction method needs to be verified and validated through computer simulations. Matlab will be used as a main tool for the implementation of the iterative reduction algorithm. Diversity of numerical examples will be used to illustrate the working ability of the algorithm. Also, the proposed technique will be compared against two popular model reduction techniques, the perturbation theory and truncation method.

4.1 Second Order System Application

Consider the following second order dynamical system in state space form. Such systems are popular mechanical systems.

$$\dot{x} = \begin{bmatrix} -2.5 & 1 \\ 1 & -1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 2 \end{bmatrix} x$$

Simulating the system with step input, then the response of the full order system is depicted an in Figure 7.

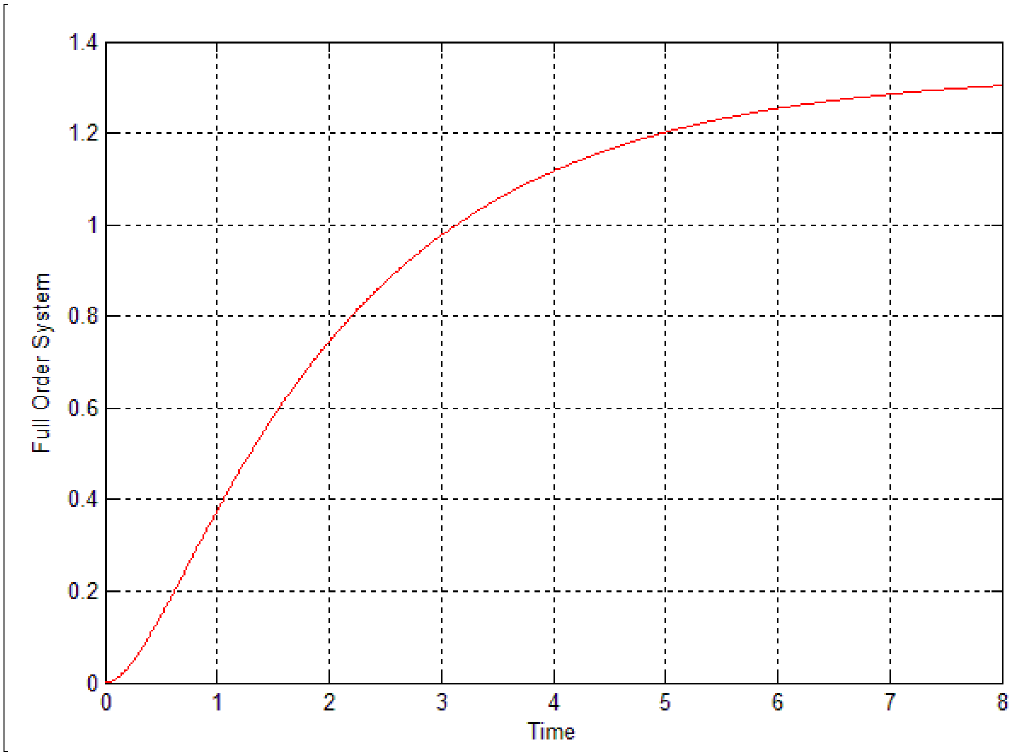


Figure 7: Full 2nd order system response

The Eigenvalues are found to be $\sigma(A) = \lambda = \{-0.5, -3\}$ making the system an over damped system.

Typically, this system has two modes; fast mode at $\lambda_2 = -3$ and slow mode at $\lambda_1 = -0.5$. Now, to find a reduced first order system with a single eigenvalue $\sigma(A_r) = -0.5$. Now, take an initial guess for A_c as *3*, so that the $\tilde{A}$ matrix becomes

$$\tilde{A} = \left[\begin{array}{c|c} -0.5 & 3 \\ \hline 0 & -3 \end{array}\right]$$

After running the algorithm (See Appendix A) the final value of A_c= *-0.1;* so that the second order system is reduced to the following system, with an eigenvalue at $\lambda = \{-0.5\}$ being preserved

$$\dot{x}_r = -0.5x_r - 3.8479\,u$$

$$y_r = -0.2077\,x_r - 0.2648\,u$$

The full order response and the reduced responses (using steepest descent) are depicted in Figure 8

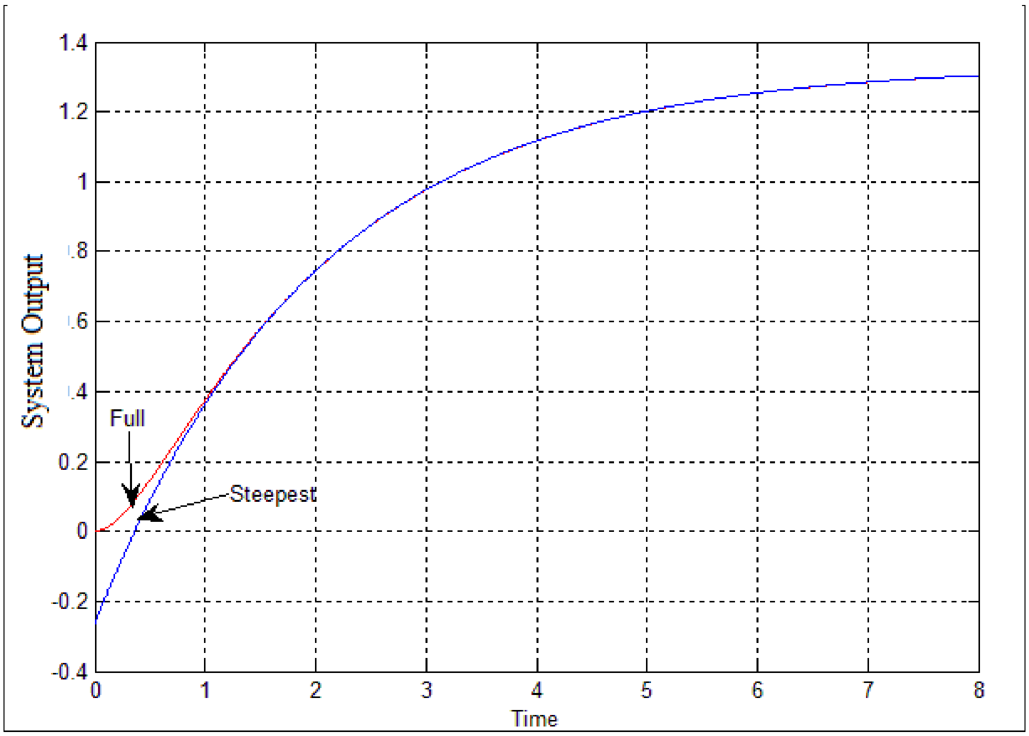


Figure 8: Full and reduced (using steepest descent) system's response

Figure 9 shows a closer window around the origin where the two responses being different.

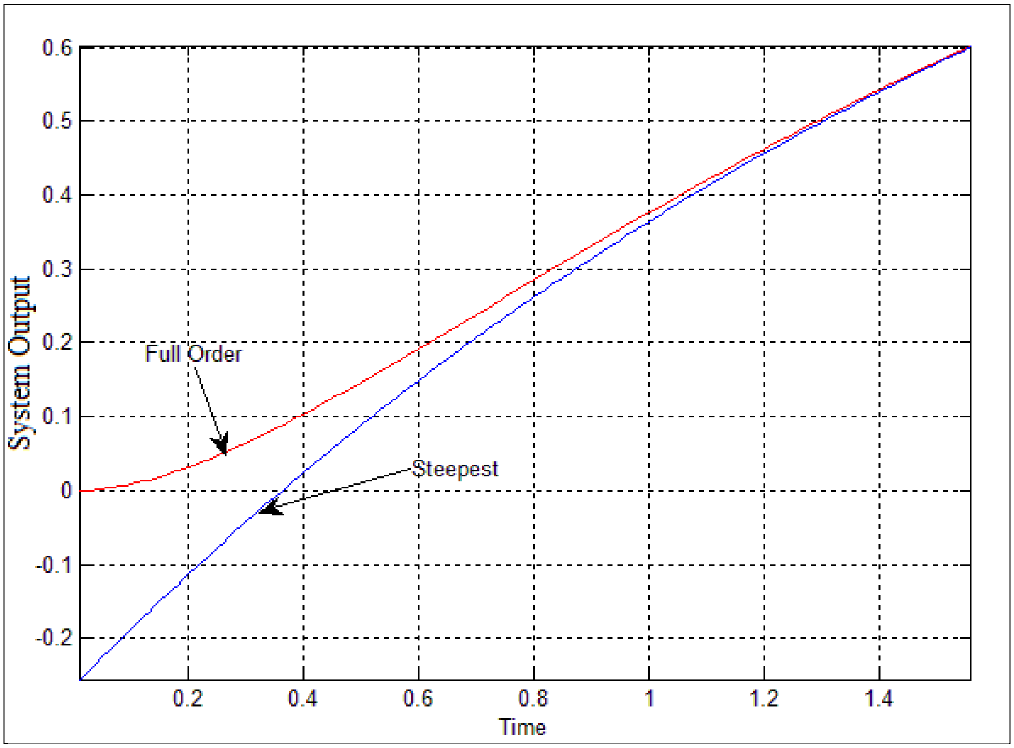


Figure 9: Full and reduced system's response for small time frame

Note that the solution is dependent on the initial guess of $A_{c,}$ so one may get results better than this for different initial guess.

The result above is compared with a Singular Perturbation method discussed in Sec.2.5; so that the original system will be reduced to

$$\dot{\bar{x}}_1 = -1.5\bar{x}_1(t) + 1u(t)$$

$$\bar{y}(t) = 2\bar{x}_1(t) + 0\,u(t)$$

It is noted that the Singular perturbation do not preserve the Eigenvalues for the original system. The eigenvalue for the reduced system is $\lambda = \{-1.5\}$. Also, Figure 10 depicts the difference in response compared with the original system. However, the steady state value (DC) for both techniques is the same.

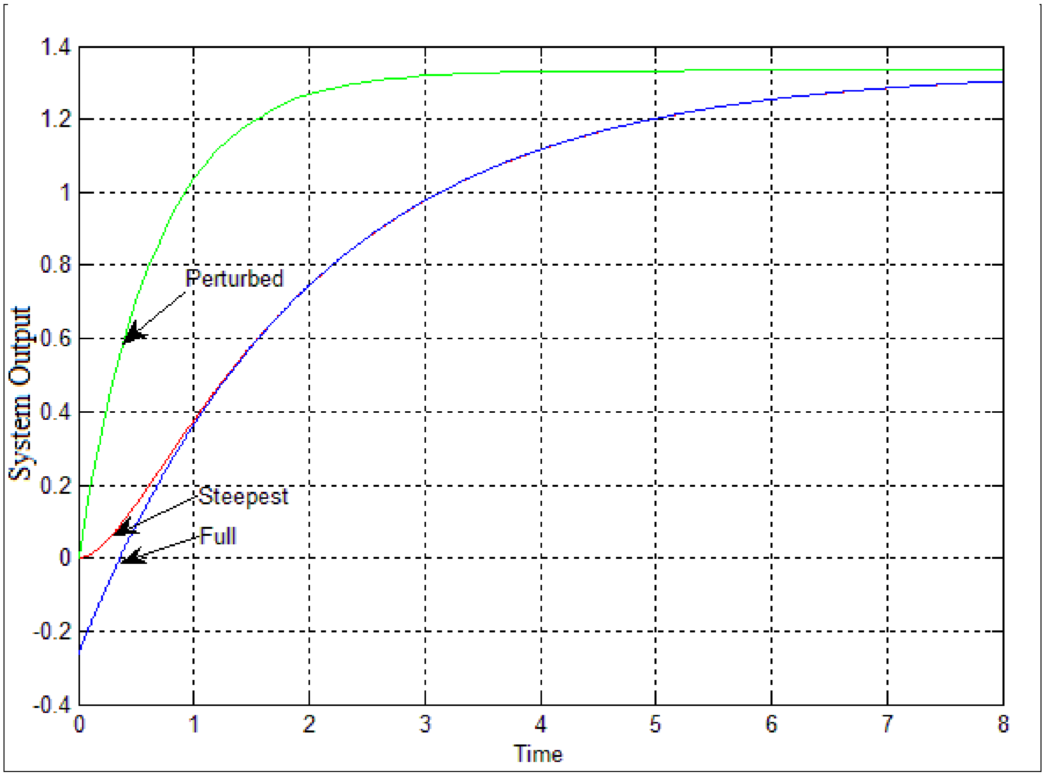


Figure 10: Full and reduced order system response for 2[nd] order system

As seen from Figure 10 that the steepest descent provided better solution than the singular perturbation, and also preserved the important Eigenvalue (i.e. frequency) for the original system. Hence, the results are more physically accepted.

4.2 Special Third Order System

Consider the following dynamical system that A_{22} matrix is not invertible

$$\dot{x} = \begin{bmatrix} -10.6 & -37.03 & -42.63 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} -0.1 \\ 4 \\ -0.3 \end{bmatrix} u$$

$$y = \begin{bmatrix} 2 & 1 & 1 \end{bmatrix} x$$

The response of the full order system is shown in Figure 11. Lets try to find a close second order system that exhibits similar dynamical behavior.

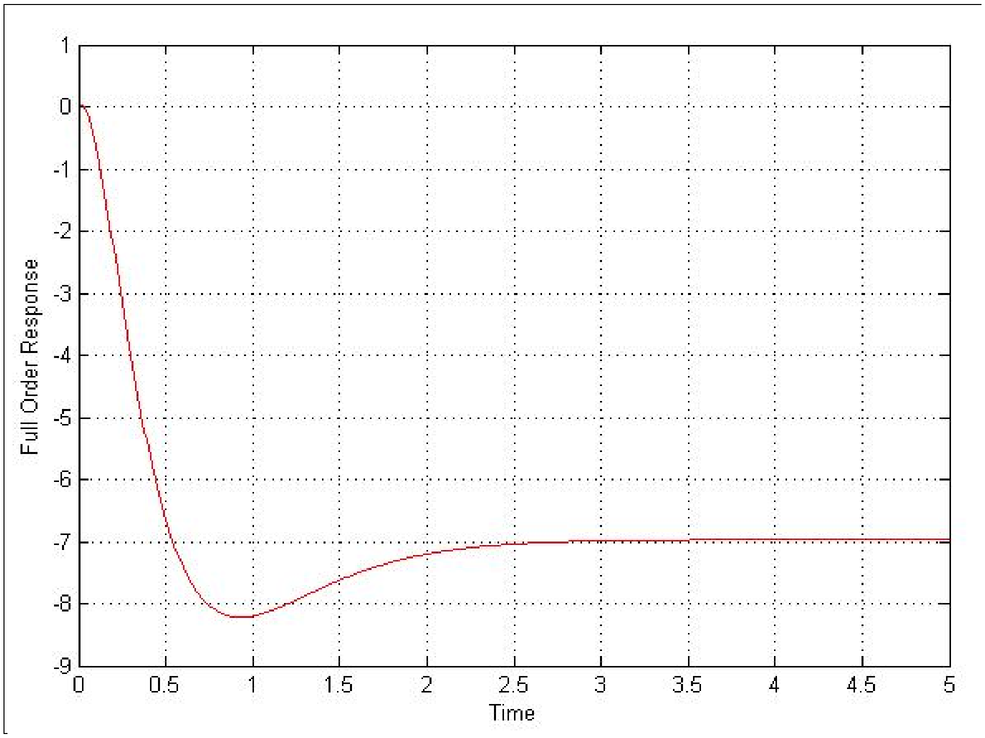


Figure 11: Full order system response for 3 × 3 system

The Eigenvalues for the full order system $\lambda = \{-2.9, -3.5, \ -4.2\}$, so it is wanted to reduce this system to 2×2 by eliminating the Eigenvalue *-4.2*. The initial guess for A_c is $[-3 \ \ 3]^T$, so the $\tilde{A}$ matrix becomes

$$\tilde{A} = \left[\begin{array}{cc|c} -2.9 & 2 & 10 \\ 0 & -3.5 & 10 \\ \hline 0 & 0 & -4.2 \end{array}\right]$$

After running the algorithm (See Appendix B) the final value of matrix A_c is given by $A_c = [-3.5357 \quad 8.1576]^T$; so that the third order system is reduced to the following system

$$\dot{x}_r = \begin{bmatrix} -2.9 & 2 \\ 0 & -3.5 \end{bmatrix} x_r + \begin{bmatrix} 8.4504 \\ -6.0075 \end{bmatrix} u$$

$$y_r = [-4.2276 \quad 0.4133]\, x_r + 1.0556\, u$$

The full order response and the reduced response are shown in Figure 12, which shows how the dynamics of the 2^{nd} order system track the full order system for a step response.

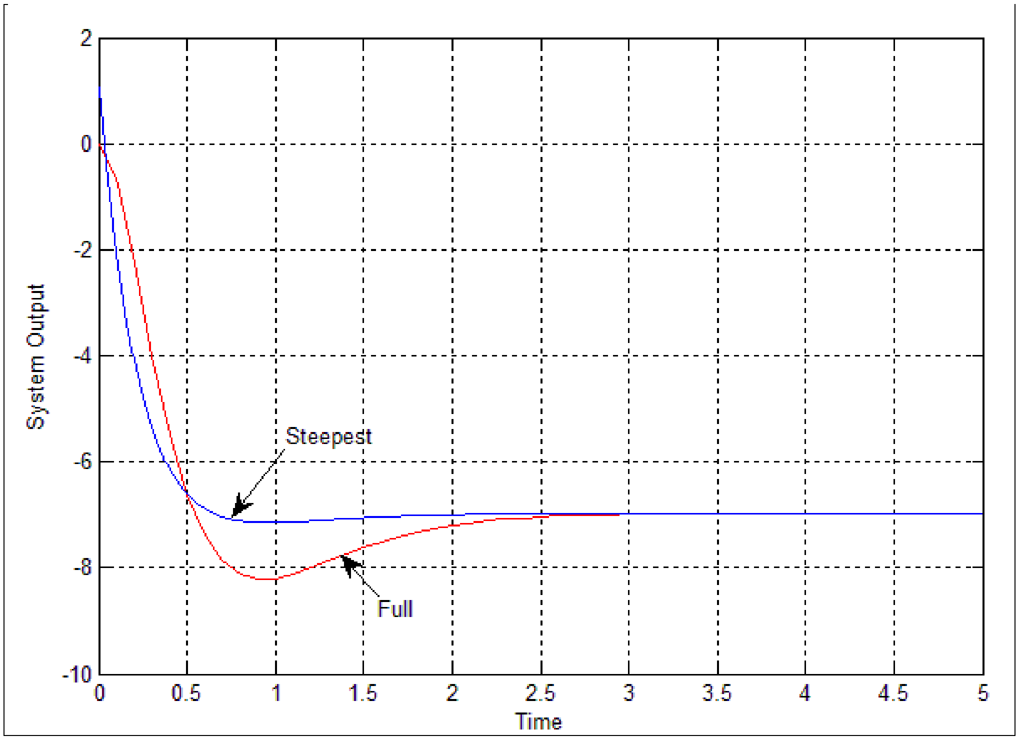


Figure 12: Full and reduced system's response using steepest method

From the original system it is noted that the A_{22} matrix is not invertible so we will compare the results with the Truncation method instead of the Permutation method.

From (Sec. 2.7) the values of R_1 and J is assumed to be

$$R_1 = \begin{bmatrix} 0 & 1 \\ 0.53 & 2 \\ 2 & 2 \end{bmatrix}$$

$$J = \begin{bmatrix} 1 & -2 & 3 \\ 3 & 5 & 4 \\ -2 & 1 & 3 \end{bmatrix}$$

So the value of T_1 is

$$T_1 = \begin{bmatrix} -1.3157 & 0.9598 \\ 0.2148 & 0.0274 \\ 0.4431 & -0.0073 \end{bmatrix}$$

And the reduced order system becomes

$$\dot{\check{x}}_1 = \begin{bmatrix} -0.2678 & 0.5634 \\ -15.0970 & -8.9035 \end{bmatrix} \check{x}_1(t) + \begin{bmatrix} 1.52 \\ 7.30 \end{bmatrix} u(t)$$

$$\check{y}(t) = [-1.9736 \quad 1.9397]\check{x}_1(t)$$

Note, the reduced system eigenvalues $\lambda = \{-1.4016 \quad -7.7697\}$ do not match with the original system's eigenvalues, that means the truncation method does not preserve the eigenvalues. Figure 13 shows the reduced system using truncation method.

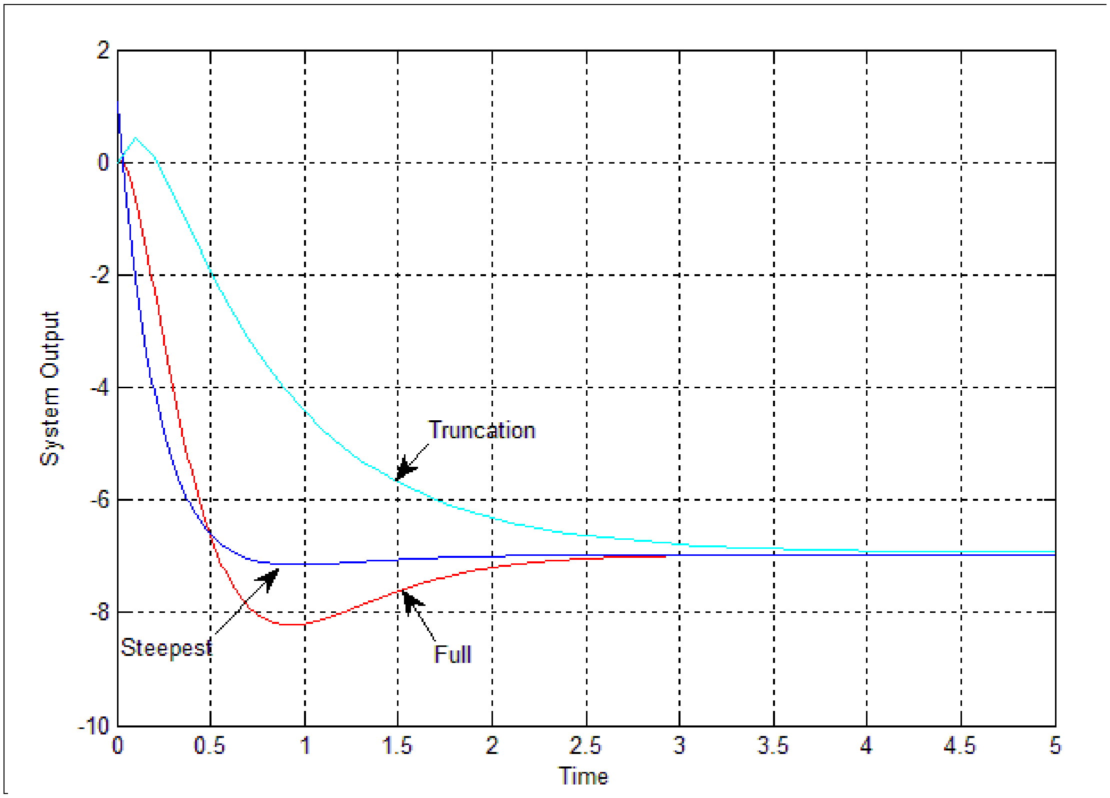


Figure 13: Full order system response using truncation method

It is noted that the response using the steepest descent method is closer to the full order response than using the truncation method.

4.3 Dynamical System with Some Complex Roots

Consider the following dynamical system that has two complex eigenvalues and one real

$$\dot{x} = \begin{bmatrix} -2 & -2.25 & -1.25 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix} x$$

The response of the full order system is shown in Figure 14. Lets try to find a close second order system that exhibits similar dynamical behavior.

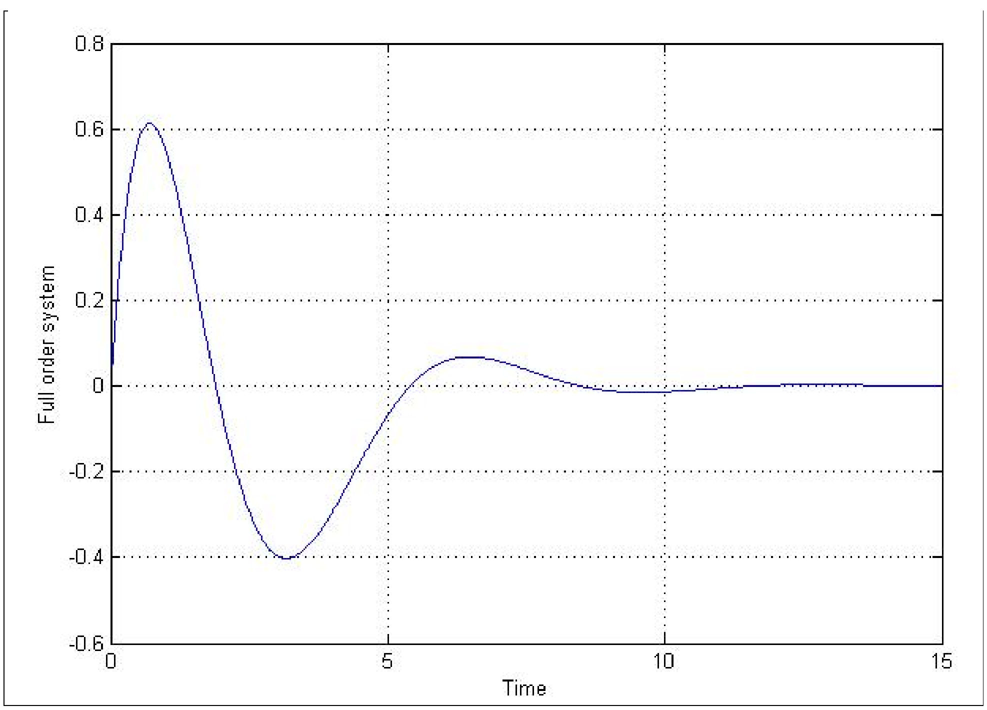


Figure 14: Full order system response for 3 × 3 system

The Eigenvalues for the full order system $\lambda = \{-0.5+i,\ -0.5-i,\ -1\}$, so it is wanted to reduce this system to 2×2 by eliminating the Eigenvalue *-1*. The initial guess for A_c is taken as $[-4 \quad 5]^T$, so the $\tilde{A}$ matrix becomes

$$\tilde{A} = \left[\begin{array}{cc|c} -1 & -1.25 & -4 \\ 1 & 0 & 5 \\ \hline 0 & 0 & -1 \end{array}\right]$$

Note that $\sigma(A_r) = \{-0.5+i,\ -0.5-i\}$.

After running the algorithm (See Appendix C where the number of iteration is 50) the final value of matrix A_c is given by $Ac = [-1.6261 \quad 1.0706]^T$; so that the third order system is reduced to the following system

$$\dot{x}_r = \begin{bmatrix} -1 & -1.25 \\ 1 & 0 \end{bmatrix} x_r + \begin{bmatrix} -0.4723 \\ -0.8927 \end{bmatrix} u$$

$$y_r = [-0.3985 \quad 0.3872]\, x_r + 0.7912\, u$$

The full order response and the reduced response are shown in Figure 18, which shows how the dynamics of the 2nd order system tracked the full order system for a step response.

From Figure 15 one can get better reduced system response by selecting another initial guess for the A_c matrix.

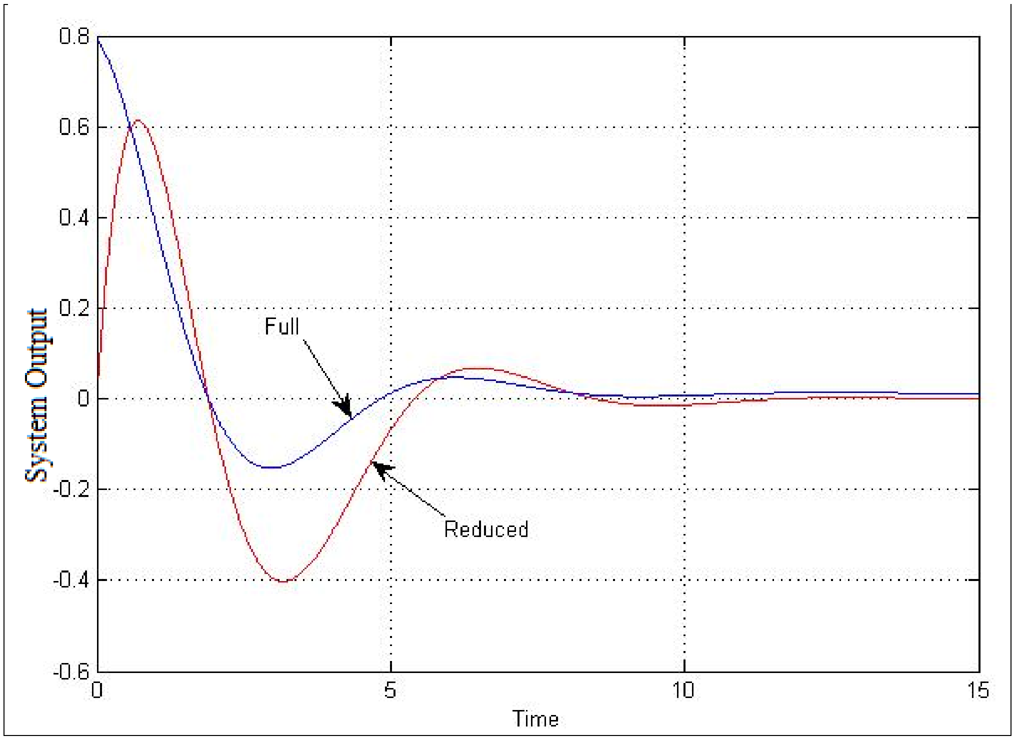


Figure 15: Full and reduced system's response using steepest method

4.4 Dynamical System with All Complex Roots

Consider the following dynamical system that

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -10 & -18 & -15 & -6 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & 1 & -1 \end{bmatrix} x$$

Lets try to find a close second order system that exhibits similar dynamical behavior.

The Eigenvalues for the full order system $\lambda = \{-1+i, -1-i, -2+i, and -2-i\}$, so it is wanted to reduce this system to 2×2 by eliminating the Eigenvalue $-2+i, and -2-i$. The initial guess for A_c is $\begin{bmatrix} 2 & -2 \\ -4 & 10 \end{bmatrix}$, so the $\tilde{A}$ matrix becomes

$$\tilde{A} = \left[\begin{array}{cc|cc} -2 & -2 & 2 & -2 \\ 1 & 0 & -4 & 10 \\ \hline 0 & 0 & -4 & -5 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

After running the algorithm (See Appendix D) the final value of matrix A_c is given by $A_c = \begin{bmatrix} 1.0893 & -0.9055 \\ 1.6290 & 5.0156 \end{bmatrix}$; so that the forth order system is reduced to the following system

$$\dot{\tilde{x}}_1 = \begin{bmatrix} -2 & 2 \\ 1 & 0 \end{bmatrix} \tilde{x}_1(t) + \begin{bmatrix} -3.7852 \\ 3.0580 \end{bmatrix} u(t)$$

$$\tilde{y}(t) = [-0.0014 \quad -1.1925]\tilde{x}_1(t) + [1.3856]u(t)$$

And the reduced system using perturbation method is given by the following form

$$\dot{\bar{x}}_1 = \begin{bmatrix} 0 & 1 \\ -0.6667 & -1.2 \end{bmatrix} \bar{x}_1(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t)$$

$$\bar{y}(t) = [-0.6667 \quad -1.2]\bar{x}_1(t) + [0]u(t)$$

Note $\sigma(A_0) = \{-0.6000 + 0.5538i \quad -0.6000 - 0.5538i\}$ do not match with the original system eigenvalues. Figure 16 shows the output for these methods

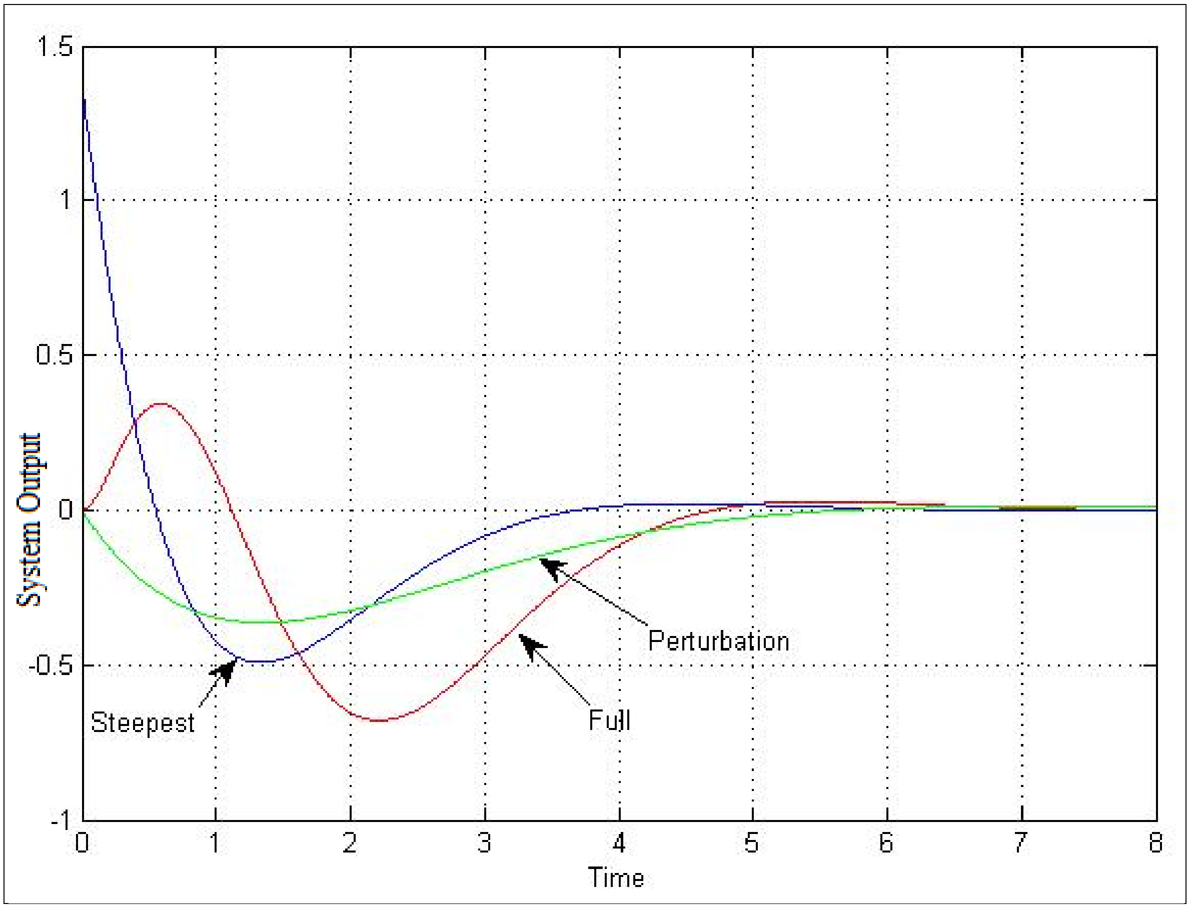


Figure 16: fourth order system with the reduction methods output

It is noted that the Steepest method preserve the important eigenvalues for the original system and from figure 16 the steepest track the full system better than perturbation.

4.5 Fifth Order System Application

In this example we want to reduce a *5×5* system to *3×3* system while preserving selected Eigenvalues. So consider the following 5th order system

$$\dot{x} = \begin{bmatrix} -15 & -85 & -225 & -274 & -120 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 3 \end{bmatrix} u$$

$$y = [0 \quad 0 \quad 0 \quad 1 \quad 2]x$$

The full order response is shown in Figure 17

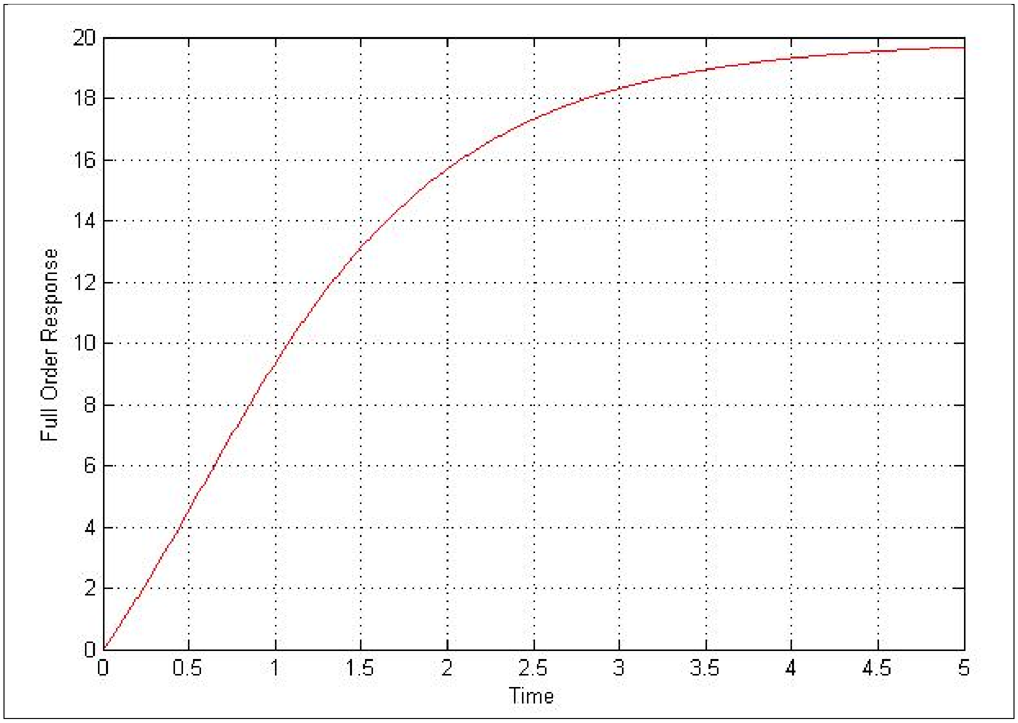


Figure 17: 5 × 5 full order system response

The Eigenvalues are found to be $\lambda = \{-1, -2, -3, -4, -5\}$, so want to reduce this system to *3×3* by eliminating the Eigenvalues *{-4, -5}*. And the initial guess for A_c is

$$Ac = \begin{bmatrix} -1 & 2 \\ 1 & 4 \\ 2 & 1 \end{bmatrix}$$

So that the $\tilde{A}$ can be selected to be

$$\tilde{A} = \left[\begin{array}{ccc|cc} -1 & -4 & 4 & -1 & 2 \\ 0 & -2 & 0 & 1 & 4 \\ 0 & 0 & -3 & 2 & 1 \\ \hline 0 & 0 & 0 & -4 & -6 \\ 0 & 0 & 0 & 0 & -5 \end{array}\right]$$

After running the algorithm (See Appendix E) the final value of A_c is

$$A_c = \begin{bmatrix} -1.0713 & 2.0883 \\ 1.2852 & 3.6466 \\ 1.4559 & 1.6791 \end{bmatrix}$$

So that the fifth order system is reduced to the following one

$$\dot{x}_r = \begin{bmatrix} -1 & -4 & 4 \\ 0 & -2 & -3 \\ 0 & 0 & -3 \end{bmatrix} x_r + \begin{bmatrix} 2863.3 \\ -346.2 \\ -174.0 \end{bmatrix} u$$

$$y_r = [0.0085 \quad -0.0340 \quad 0.0358]\, x_r - 6.2386\, u$$

The full order response and the reduced response are placed on one Figure 18

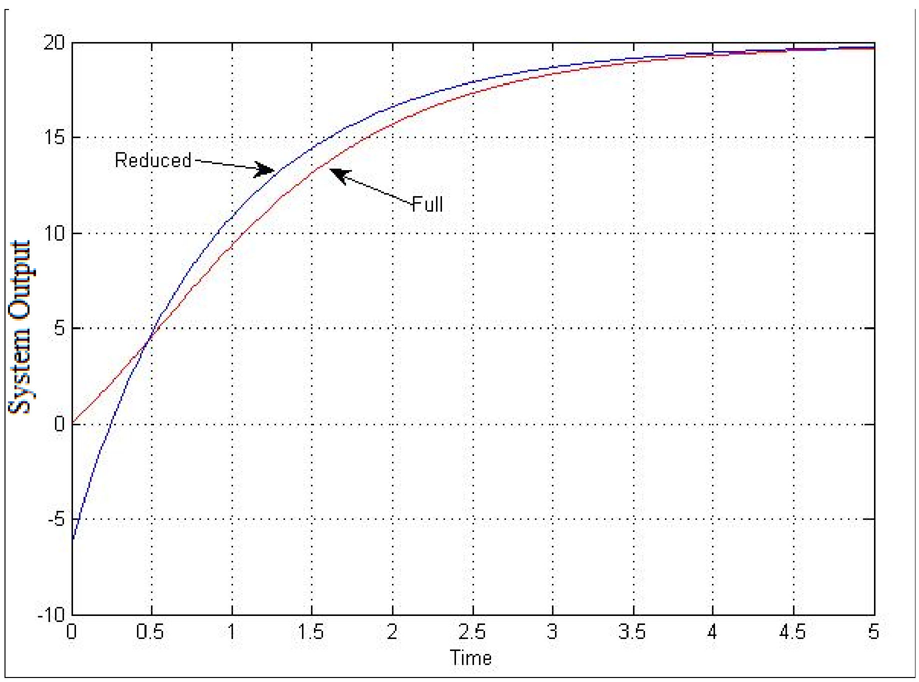


Figure 18: Full and reduced system's response using steepest method

Let us now compare the steepest descent method with the singular perturbation method for second and third order systems.

As in previous example the A_{22} matrix for the original system is not invertible so to compare the results the Truncation method is used instead of Permutation method.

From (Sec. 2.7) the values of R_1 and J is assumed to be

$$R_1 = \begin{bmatrix} 1 & 4 & -1 \\ 1 & 3 & -2 \\ 2 & 2 & -1 \\ 2 & -1 & 2 \\ 1 & 1 & -2 \end{bmatrix}$$

$$J = \begin{bmatrix} -1 & 2 & -3 & 4 & 5 \\ 1 & -2 & 3 & -4 & 6 \\ 1 & -2 & 3 & 4 & 7 \\ 1 & 2 & 3 & 4 & -8 \\ -1 & 2 & 3 & -4 & 9 \end{bmatrix}$$

So the value of T_1 is

$$T_1 = \begin{bmatrix} 0.0120 & 0.0687 & 0.0481 \\ 0.0291 & 0.0241 & 0.0453 \\ -.0667 & 0.4742 & 0.1608 \\ 0.3647 & -.2485 & 0.0782 \\ 0.3630 & -.5441 & -.5715 \end{bmatrix}$$

And the reduced order system is

$$\dot{\breve{x}}_1 = \begin{bmatrix} -130.8238 & 24.4400 & 6.9214 \\ -523.9753 & 94.0247 & 25.6835 \\ 130.2093 & -22.3393 & -6.3590 \end{bmatrix} \breve{x}_1(t) + \begin{bmatrix} 11 \\ 10 \\ -6 \end{bmatrix} u(t)$$

$$\breve{y}(t) = [1.0907 \quad -1.3367 \quad -1.0647] \breve{x}_1(t)$$

So the response for the truncation method is shown in Figure 19

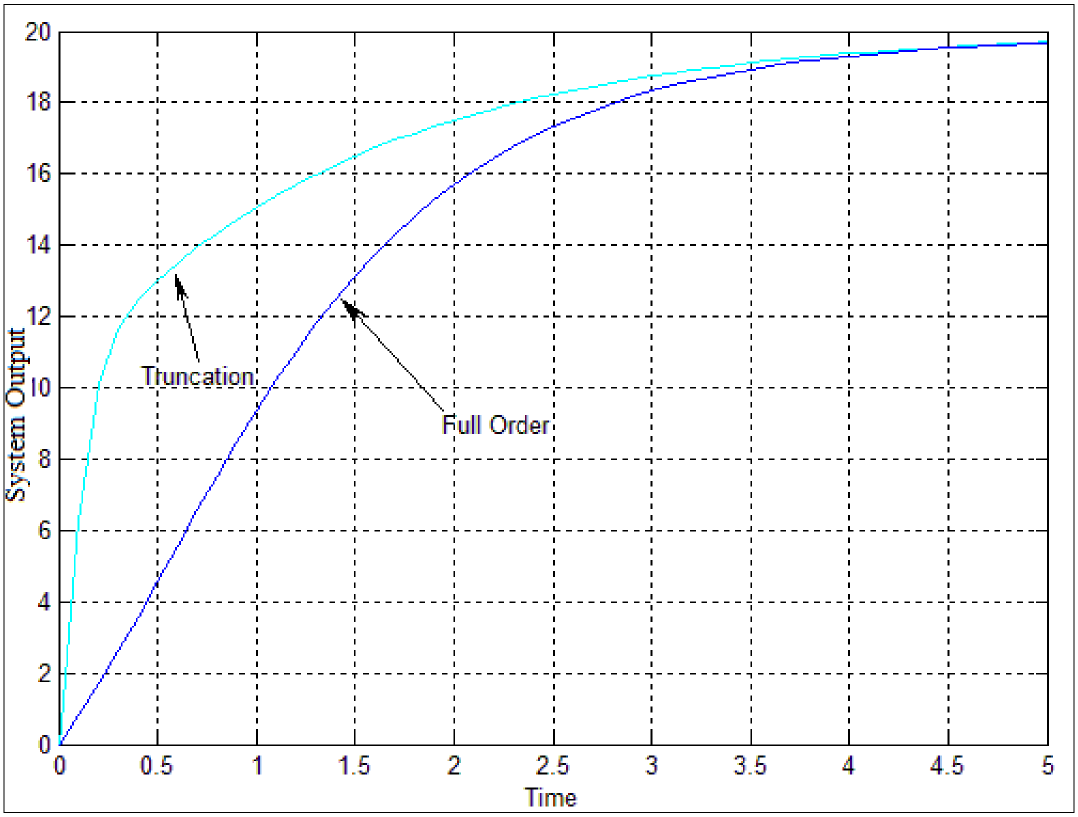


Figure 19: Full order and reduced order (truncation method) for 5×5 system reduced to 3×3

It is noted that the response using the steepest descent method is closer to the full order response than using the truncation method.

Chapter Five

Conclusions and Future Work

Steepest descent search method was used successfully in optimizing the reducible matrix reduction technique coupling term.

The optimization solution was compared with perturbation theory based reduction and the Truncation method. The proposed techniques seem to work nicely and it resulted in superior responses.

Finally, the proposed technique had a major advantage in preserving selected frequencies (eigenvalues) making the reduced system more physically accepted.

Closed form optimized solution for the reducible matrix model reduction method was not possible. The complex and rigorous mathematics that are needed for the closed solution required advance knowledge in calculus of variation. This part is a good future work for a PhD dissertation.

REFERENCES

Abdalla M., and Alsmadi O. (2006), **Model Reduction via Reducibility Matrix**, Journal of Donghua University, 23, 6.

Abdalla M. (2009), **Model Reduction via Particle Swarm Optimization (PSO)**, IASTED Proceedings: Canada.

Anton H., (2005), **Elementary Linear Algebra,** John Wiley and Sons.

Cunedioglu Yusuf, Mugan Ata, and Akcay Huseyin (2005), **Frequency domain analysis of model order model reduction technique**, Finite Element in Analysis and Design, 42, 367-403.

Davison E. J(1966). **A method for simplifying linear dynamic systems**. IEEE Trans. Autom. Contr., 11:93–101.

Edward L. Wilson, M. ASCE, and Eduardo P. Bayo (1986), **Use of special Ritz vector in dynamic substructure analysis**, Vol. 112, No. 8, pp. 1944-1954.

Gyuan R.J. (1965). **Reduction of stiffness and mass matrices**. AIAA Journal, 3, 280.

Graham A. (1981), **Kronecker Products and Matrix Calculus: with Applications**, Ellis Horwood Limited.

Irons B.M. (1965). **Structural Eigenvalue problems: elimination of unwanted variables**. AIAA Journal, 3, 961-962.

Kammer D.C. (1987), **Test-analysis model development using an exact modal reduction**, International Journal of Analytical and Experimental Modal Analysis. p. 174.

Ogata K. (2002), **Modren Control Engineering**, 4th edition, Person Education International, Prentice Hall.

O'Callahan, J. C., Avitabile P. A., and Riemer R. (1987), **System equivalent reduction expansion process**, in proceeding of the 7th international modal analysis conference, Las Vegas, Nevada, p.29.

Papadopulos Michael, and Garcia Ephrahim (1996), **Improvement of model reduction schemes using the system equivalent reduction expansion process**, AIAA JOURNAL: USA, 34, 10.

Sellgren U. (2003), **Component mode synthesis - a method for efficient dynamic simulation of complex technical systems**, Technical Report, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden.

Skelton R. E., Oliveira M. C., and Han J. H., **System modeling and model reduction.**

Syrmos V. L. and Lewis F. L. (1995), **Optimal Control**, 2nd edition, Wiley-Interscience Publication.

Varga, A. (1994), **On Modal Techniques for Model Reduction**. In: Proc. 1st IMACS Conference on Mathematical Modelling, MATHMOD , vol. 2 , pp. 198-202.

APPENDIX A: MATLAB CODE FOR REDUCING 2×2 SYSTEM INTO 1×1 USING STEEPESTDESCENT AND SINGULAR PERTURBATION

```
%%                      Assume Ac.
%                   Find P using LMI.
%                 Find x and xr values.
%                    Check the norm.
% Find new Ac using Steepest Descent (Assume Ac=const.).
%             Ac=Ac-alpha*(gradient of F).
%                        New Loop.

clc
close all
clear all
syms a0;
r=1; o=1;
n=r+o;
tf=8;
t=0:0.01:tf;
A=[-2.5   1;...
    1     -1];
B=[1; 0];
C=[0 2] ;
D=0 ;
u=1;

x0=[0 0]';
At=[-0.5     3;...
     0    -3];

Ar=At(1:r,1:r);
Ao=At(1+r:n,1+r:n);
A0=At(1+r:n,1:r);
Ac=At(1:r,1+r:n);
Oo=zeros(o,o); Or=zeros(r,r); Oro=zeros(r,o); Oor=zeros(o,r);
K6=[eye(r) Oro];
K7=[Oor eye(o)];

sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
P=eye(n);
E(2)=100;
E(1)=200;
k=2;
while E(k)>1e-3
    k=k+1;
    Pnew=P;

%% LMI
        for i=1:2
            At=[Ar Ac;...
                A0 Ao];
            options=[0 0 -1 0 0] ;
            I=eye(2) ;
            W=diag(diag(At)) ;
```

```
    Po=W ;
    e=1e-12 ;

    setlmis([]);
    S=lmivar(1,[2,1]);
    P=lmivar(2,[2,2]);

    lmiterm([-1 1 1 S],1,1);        % LMI #1: S
    lmiterm([-1 2 1 -P],1,1);       % LMI #1: P'
    lmiterm([-1 2 1 0],-Po');       % LMI #1: -Po'
    lmiterm([-1 2 2 0],I);          % LMI #1: I

    lmiterm([-2 1 1 0],e*I);        % LMI #2: e*I
    lmiterm([-2 2 1 -P],1,A');      % LMI #2: P'*A'
    lmiterm([-2 2 1 -P],At',-1);    % LMI #2: -At'*P'
    lmiterm([-2 2 2 0],I);          % LMI #2: I
    lmisys=getlmis;

    %---Objective Function Definition---%
    n=decnbr(lmisys) ;
    c=zeros(n,1) ;
    for j=1:n
        %Ei=defcx(lmisys,j,e) ;
        %c(j)=Ei ;
        Si=defcx(lmisys,j,S) ;
        c(j)=trace(Si) ;
    end ;
    %---Lmi Solution---%
    t0=cputime ;
    [copt,popt]=mincx(lmisys,c,options) ;
    %Gama=sqrt(dec2mat(lmisys,popt,e))
    Popt=dec2mat(lmisys,popt,P);
end
%%
n=2;
P=Popt;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
E(k)=norm(y-y1)
if E(k)-E(k-1)>=0
    break
else
    Acnew=Ac;
    %% New Value for Ac
    syms ts
    syms ac1
    Acn=[ac1];
```

```
            I=-Acn*inv(Ao)*Bo*u+Br*u;
            xrs=expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-inv(Ar)*I;
            xs=expm(A*ts)*(x0+inv(A)*B)-inv(A)*B;
            f1=(C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2;
            f=sqrt(int(f1,ts,0,tf));
            gf(1)=subs(diff(f,ac1),{ac1},{Ac(1)});
%             Ac=Ac-.1*gf'/norm(gf(:))
            Ac2=Ac-a0*gf'/norm(gf(:));
            I=eval(eval(-Ac2*inv(Ao)*Bo*u+Br*u));
            xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-
inv(Ar)*I));
            xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
            f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2));
            f=eval(eval(sqrt(int(f1,0,tf))));
            alpha=solve(eval(diff(f,a0)),a0);
            Ac=eval(Ac-alpha(1)*gf'/norm(gf(:)))
        end
end
Ac=Acnew
P=Pnew;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
plot(t,y,'r',t1,y1)
hold on
%% Singular Perturbation
n=r+o;
Ap=[-2.5   1;...
    1     -1];
Bp=[1; 0];
Cp=[0 2] ;
Dp=0 ;
up=1;

A11=Ap(1:r,1:r);
A12=Ap(1:r,1+r:n);
A21=Ap(1+r:n,1:r);
A22=Ap(1+r:n,1+r:n);
B1=Bp(1:r,1);
B2=Bp(1+r:n,1);
C1=Cp(1,1:r);
C2=Cp(1,1+r:n);

A0=A11-A12*inv(A22)*A21;
B0=B1-A12*inv(A22)*B2;
C0=C1-C2*inv(A22)*A21;
D0=-C2*inv(A22)*B2;
sys=ss(A0,B0,C0,D0);
[y2,t2,x]=step(sys,t);
```

```
plot(t2,y2,'g')
hold on
grid
%% Truncation
t=0:0.1:tf;
R1=[-1.2; -3]; J=[1 2; 3 5];
T1=J'*R1*inv(R1'*J'*R1);
Br=R1'*B;Ar=R1'*A*T1;
Br=R1'*B;
Cr=C*T1;
Dr=D;
sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
sys2=ss(Ar,Br,Cr,Dr);
[y2,t2,x2] = step(sys2,t);  %solve x
plot(t2,y2,'c')
PP=R1*T1';
grid
```

APPENDIX B: MATLAB CODE FOR REDUCING 3×3 SYSTEM INTO 2×2 USING STEEPEST DESCENT AND TRUNCATION

```
%%                      Assume Ac.
%                    Find P using LMI.
%                  Find x and xr values.
%                     Check the norm.
% Find new Ac using Steepest Descent (Assume Ac=const.).
%              Ac=Ac-alpha*(gradient of F).
%                         New Loop.

clc
close all
clear all
syms a0;
r=2; o=1;
n=r+o;
tf=5;
t=0:0.01:tf;
A=[-10.6 -37.03 -42.63 ;
   1      0       0 ;
   0     1       0] ;
B=[-.1; 4; -.3];
C=[2 1 1] ;
D=0 ;
u=1;
x0=[0 0 0]';
At=[-2.9   2    -3;...
      0  -3.5   3;...
      0    0  -4.2];
Ar=At(1:r,1:r);
Ao=At(1+r:n,1+r:n);
A0=At(1+r:n,1:r);
Ac=At(1:r,1+r:n);
Oo=zeros(o,o); Or=zeros(r,r); Oro=zeros(r,o); Oor=zeros(o,r);
K6=[eye(r) Oro];
K7=[Oor eye(o)];

sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
E(2)=1;
E(1)=2;
E(2)=100;
E(1)=200;
k=2;
while E(k)>1e-3
    k=k+1;
    %% LMI
    for i=1:3
        At=[Ar Ac;...
            A0 Ao];

        options=[0 0 -1 0 0] ;
        I=eye(3) ;
        W=diag(diag(At)) ;
```

```
    Po=W ;
    e=1e-12 ;

    setlmis([]);
    S=lmivar(1,[3,1]);
    P=lmivar(2,[3,3]);

    lmiterm([-1 1 1 S],1,1);                % LMI #1: S
    lmiterm([-1 2 1 -P],1,1);               % LMI #1: P'
    lmiterm([-1 2 1 0],-Po');               % LMI #1: -Po'
    lmiterm([-1 2 2 0],I);                  % LMI #1: I

    lmiterm([-2 1 1 0],e*I);                % LMI #2: e*I
    lmiterm([-2 2 1 -P],1,A');              % LMI #2: P'*A'
    lmiterm([-2 2 1 -P],At',-1);            % LMI #2: -At'*P'
    lmiterm([-2 2 2 0],I);                  % LMI #2: I
    lmisys=getlmis;

    %---Objective Function Definition---%
    n=decnbr(lmisys) ;
    c=zeros(n,1) ;
    for j=1:n
        %Ei=defcx(lmisys,j,e) ;
        %c(j)=Ei ;
        Si=defcx(lmisys,j,S) ;
        c(j)=trace(Si) ;
    end ;
    %---Lmi Solution---%
    t0=cputime ;
    [copt,popt]=mincx(lmisys,c,options) ;
    %Gama=sqrt(dec2mat(lmisys,popt,e))
    Popt=dec2mat(lmisys,popt,P);
end
%%
n=3;
P=Popt;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
E(k)=norm(y-y1)
if E(k)-E(k-1)>=0
    break
else
    %% New Value for Ac
    syms ts
    syms ac1 ac2
    Acn=[ac1; ac2];
    -C*P*K6'*xrs+C*P*K7'*inv(Ao)*K7*inv(P)*B*u)^2,ts,0,3));
    I=eval(eval(-Acn*inv(Ao)*Bo*u+Br*u));
```

```
        xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-inv(Ar)*I));
        xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
        f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2))
        f=eval(eval(sqrt(int(f1,0,tf))))
        gf(1)=subs(diff(f,ac1),{ac1 ac2},{Ac(1) Ac(2)});
        gf(2)=subs(diff(f,ac2),{ac1 ac2},{Ac(1) Ac(2)});
%         Ac=Ac-0.008*gf'/norm(gf(:))
        Ac2=Ac-a0*gf'/norm(gf(:));
        I=eval(eval(-Ac2*inv(Ao)*Bo*u+Br*u));
        xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-inv(Ar)*I));
        xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
        f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2));
        f=eval(eval(sqrt(int(f1,0,tf))));
        alpha=solve(eval(diff(f,a0)),a0);
        Ac=eval(Ac-alpha(1)*gf'/norm(gf(:)))
    end
end
plot(t,y,'r',t1,y1)

%% Truncation
t=0:0.1:tf;
R1=[0 1; 0.53 2; 2 2]; J=[1 -2 3; 3 5 4; -2 1 3];
T1=J'*R1*inv(R1'*J'*R1);
Art=R1'*A*T1;
Brt=R1'*B;
Crt=C*T1;
Drt=D;
sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
sys2=ss(Art,Brt,Crt,Drt);
[y2,t2,x2] = step(sys2,t);  %solve x
plot(t2,y2,'c')
PP=R1*T1';
grid
```

APPENDIX C: MATLAB CODE FOR REDUCING 3×3 SYSTEM INTO 2×2 USING STEEPEST DESCENT WITH COMPLEX EIGENVALUES

```
%%                      Assume Ac.
%                   Find P using LMI.
%                 Find x and xr values.
%                    Check the norm.
% Find new Ac using Steepest Descent (Assume Ac=const.).
%             Ac=Ac-alpha*(gradient of F).
%                        New Loop.

clc
close all
clear all
syms a0;
r=2; o=1;
n=r+o;
tf=15;
t=0:0.01:tf;
A=[-2.0000   -2.2500   -1.2500
    1.0000         0         0
         0    1.0000         0];
B=[1;0;0];
C=[2 0 0] ;
D=0;
u=1;
x0=[0 0 0]';
At=[-1   -1.25    -4;...
      1    0       5;...
      0    0      -1];
Ar=At(1:r,1:r);
Ao=At(1+r:n,1+r:n);
A0=At(1+r:n,1:r);
Ac=At(1:r,1+r:n);
Oo=zeros(o,o); Or=zeros(r,r); Oro=zeros(r,o); Oor=zeros(o,r);
K6=[eye(r) Oro];
K7=[Oor eye(o)];

sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
P=eye(n);
E(2)=100;
E(1)=200;
k=2;
while E(k)>1e-3
    k=k+1;
        Pnew=P;
    %% LMI
    for i=1:3
        At=[Ar Ac;...
            A0 Ao];

        options=[0 0 -1 0 0] ;
        I=eye(3) ;
        W=diag(diag(At)) ;
```

```
    Po=W ;
    e=1e-12 ;

    setlmis([]);
    S=lmivar(1,[3,1]);
    P=lmivar(2,[3,3]);

    lmiterm([-1 1 1 S],1,1);      % LMI #1: S
    lmiterm([-1 2 1 -P],1,1);     % LMI #1: P'
    lmiterm([-1 2 1 0],-Po');     % LMI #1: -Po'
    lmiterm([-1 2 2 0],I);        % LMI #1: I

    lmiterm([-2 1 1 0],e*I);      % LMI #2: e*I
    lmiterm([-2 2 1 -P],1,A');    % LMI #2: P'*A'
    lmiterm([-2 2 1 -P],At',-1);  % LMI #2: -At'*P'
    lmiterm([-2 2 2 0],I);        % LMI #2: I
    lmisys=getlmis;

    %---Objective Function Definition---%
    n=decnbr(lmisys) ;
    c=zeros(n,1) ;
    for j=1:n
        %Ei=defcx(lmisys,j,e) ;
        %c(j)=Ei ;
        Si=defcx(lmisys,j,S) ;
        c(j)=trace(Si) ;
    end ;
    %---Lmi Solution---%
    t0=cputime ;
    [copt,popt]=mincx(lmisys,c,options) ;
    %Gama=sqrt(dec2mat(lmisys,popt,e))
    Popt=dec2mat(lmisys,popt,P);
end
    Acnew=Ac;
%%
n=3;
P=Popt;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
E(k)=norm(y-y1)
if E(k)-E(k-1)>=0
    break
else
    %% New Value for Ac
    syms ts
    syms ac1 ac2
    Acn=[ac1; ac2];
    I=eval(eval(-Acn*inv(Ao)*Bo*u+Br*u));
```

```
        xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-inv(Ar)*I));
        xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
        f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2))
        f=eval(eval(sqrt(int(f1,0,tf))))
        gf(1)=subs(diff(f,ac1),{ac1 ac2},{Ac(1) Ac(2)});
        gf(2)=subs(diff(f,ac2),{ac1 ac2},{Ac(1) Ac(2)});
%         Ac=Ac-.1*gf'/norm(gf(:))
        Ac2=Ac-a0*gf'/norm(gf(:));
        I=eval(eval(-Ac2*inv(Ao)*Bo*u+Br*u));
        xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-inv(Ar)*I));
        xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
        f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2));
        f=eval(eval(sqrt(int(f1,0,tf))));
        alpha=solve(eval(diff(f,a0)),a0);
        Ac=eval(Ac-alpha(1)*gf'/norm(gf(:)))
    end
end
Ac=Acnew
P=Pnew;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);

plot(t,y,'r',t1,y1)
hold on
```

APPENDIX D: MATLAB CODE FOR REDUCING 4×4 SYSTEM INTO 2×2 USING STEEPEST DESCENT AND PERTURBATION METHOD WITH ALL COMPLEX EIGENVALUES

```
%%                      Assume Ac.
%                  Find P using LMI.
%                Find x and xr values.
%                   Check the norm.
% Find new Ac using Steepest Descent (Assume Ac=const.).
%             Ac=Ac-alpha*(gradient of F).
%                       New Loop.

clc
close all
clear all
syms a0
r=2; o=2;
n=r+o;
tf=8;
t=0:0.01:tf;
A=[  0     1     0     0;
     0     0     1     0;
     0     0     0     1;
   -10   -18   -15   -6];
B=[1;0;0;0];
C=[0 0 1 -1] ;
D=0;
u=1;
x0=[0 0 0 0]';
At=[-2    -2     2     -2;
     1     0    -4     10;
     0     0    -4     -5;
     0     0     1      0];
Ar=At(1:r,1:r);
Ao=At(1+r:n,1+r:n);
A0=At(1+r:n,1:r);
Ac=At(1:r,1+r:n);
Oo=zeros(o,o); Or=zeros(r,r); Oro=zeros(r,o); Oor=zeros(o,r);
K6=[eye(r) Oro];
K7=[Oor eye(o)];

sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
P=eye(n);
E(2)=40;
E(1)=50;
k=2;
while E(k)>1e-3
    k=k+1;
    Pnew=P;
%% LMI
        for i=1:4
            At=[Ar Ac;...
                A0 Ao];

            options=[0 0 -1 0 0] ;
```

```
    I=eye(4) ;
    W=diag(diag(At)) ;

    Po=W ;
    e=1e-12 ;

    setlmis([]);
    S=lmivar(1,[4,1]);
    P=lmivar(2,[4,4]);

    lmiterm([-1 1 1 S],1,1);                           % LMI #1: S
    lmiterm([-1 2 1 -P],1,1);                          % LMI #1: P'
    lmiterm([-1 2 1 0],-Po');                          % LMI #1: -
Po'
    lmiterm([-1 2 2 0],I);                             % LMI #1: I

    lmiterm([-2 1 1 0],e*I);                           % LMI #2: e*I
    lmiterm([-2 2 1 -P],1,A');                         % LMI #2:
P'*A'
    lmiterm([-2 2 1 -P],At',-1);                       % LMI #2: -
At'*P'
    lmiterm([-2 2 2 0],I);                             % LMI #2: I
    lmisys=getlmis;

    %---Objective Function Definition---%
    n=decnbr(lmisys) ;
    c=zeros(n,1) ;
    for j=1:n
        %Ei=defcx(lmisys,j,e) ;
        %c(j)=Ei ;
        Si=defcx(lmisys,j,S) ;
        c(j)=trace(Si) ;
    end ;
    %---Lmi Solution---%
    t0=cputime ;
    [copt,popt]=mincx(lmisys,c,options) ;
    %Gama=sqrt(dec2mat(lmisys,popt,e))
    Popt=dec2mat(lmisys,popt,P);
end
%%
n=4;
P=Popt;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
E(k)=norm(y-y1)
if E(k)-E(k-1)>=0
    break
else
```

```
            Acnew=Ac;
            %% New Value for Ac
            syms ts
            syms ac1 ac2 ac3 ac4
            Acn=[ac1 ac3; ac2 ac4];
            I=eval(eval(-Acn*inv(Ao)*Bo*u+Br*u));
            xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-
inv(Ar)*I));
            xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
            f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2))
            f=eval(eval(sqrt(int(f1,0,tf))))

            gf(1,1)=subs(eval(eval(diff(f,ac1))),...
                {ac1 ac2 ac3 ac4},{Ac(1) Ac(2) Ac(3) Ac(4)});
            gf(2,1)=subs(eval(eval(diff(f,ac2))),...
                {ac1 ac2 ac3 ac4},{Ac(1) Ac(2) Ac(3) Ac(4)});
            gf(1,2)=subs(eval(eval(diff(f,ac3))),...
                {ac1 ac2 ac3 ac4},{Ac(1) Ac(2) Ac(3) Ac(4)});
            gf(2,2)=subs(eval(eval(diff(f,ac4))),...
                {ac1 ac2 ac3 ac4},{Ac(1) Ac(2) Ac(3) Ac(4)});
%             Ac=Ac-1*gf/norm(gf(:))
            Ac2=Ac-a0*gf/norm(gf(:));
            I=eval(eval(-Ac2*inv(Ao)*Bo*u+Br*u));
            xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-
inv(Ar)*I));
            xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
            f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2));
            f=eval(eval(sqrt(int(f1,0,tf))));
            alpha=solve(eval(diff(f,a0)),a0);
            Ac=eval(Ac-alpha(1)*gf/norm(gf(:)))
        end
end
Ac=Acnew
P=Pnew;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
plot(t,y,'r',t1,y1)
hold on
grid
%% Singular Perturbation
n=r+o;
Ap=[  0     1     0     0;
     0     0     1     0;
     0     0     0     1;
   -10   -18   -15    -6];
Bp=[1;0;0;0];
Cp=[0 0 1 -1] ;
Dp=0;
```

```
A11=Ap(1:r,1:r);
A12=Ap(1:r,1+r:n);
A21=Ap(1+r:n,1:r);
A22=Ap(1+r:n,1+r:n);
B1=Bp(1:r,1);
B2=Bp(1+r:n,1);
C1=Cp(1,1:r);
C2=Cp(1,1+r:n);

A0=A11-A12*inv(A22)*A21;
B0=B1-A12*inv(A22)*B2;
C0=C1-C2*inv(A22)*A21;
D0=-C2*inv(A22)*B2;
sys=ss(A0,B0,C0,D0);
[y2,t2,x]=step(sys,t);
plot(t2,y2,'g')
hold on
grid
```

APPENDIX E: MATLAB CODE FOR REDUCING 5×5 SYSTEM INTO 3×3 USING STEEPEST DESCENT AND TRUNCATION

```
%%                      Assume Ac.
%                   Find P using LMI.
%                 Find x and xr values.
%                    Check the norm.
% Find new Ac using Steepest Descent (Assume Ac=const.).
%             Ac=Ac-alpha*(gradient of F).
%                        New Loop.

clc
close all
clear all
syms a0
r=3; o=2;
n=r+o;
tf=5;
t=0:0.01:tf;
A=[-15   -85  -225  -274  -120
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0];
B=[  1;
     1;
     1;
     2;
     3];
C=[0     0     0      1     2] ;
D=0;
u=1;
x0=[0 0 0 0 0]';
At=[-1    -4     4     -1     2;
     0    -2    -3      1     4;
     0     0    -3      2     1;
     0     0     0     -4    -6;
     0     0     0      0    -5];
Ar=At(1:r,1:r);
Ao=At(1+r:n,1+r:n);
A0=At(1+r:n,1:r);
Ac=At(1:r,1+r:n);
Oo=zeros(o,o); Or=zeros(r,r); Oro=zeros(r,o); Oor=zeros(o,r);
K6=[eye(r) Oro];
K7=[Oor eye(o)];

sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
P=eye(n);
E(2)=1000;
E(1)=2000;
k=2;
while E(k)>1e-3
    k=k+1;
    Pnew=P;
```

```
%% LMI
        for i=1:5
            At=[Ar Ac;...
                A0 Ao];

            options=[0 0 -1 0 0] ;
            I=eye(5) ;
            W=diag(diag(At)) ;

            Po=W ;
            e=1e-12 ;

            setlmis([]);
            S=lmivar(1,[5,1]);
            P=lmivar(2,[5,5]);

            lmiterm([-1 1 1 S],1,1);        % LMI #1: S
            lmiterm([-1 2 1 -P],1,1);       % LMI #1: P'
            lmiterm([-1 2 1 0],-Po');       % LMI #1: -Po'
            lmiterm([-1 2 2 0],I);          % LMI #1: I

            lmiterm([-2 1 1 0],e*I);        % LMI #2: e*I
            lmiterm([-2 2 1 -P],1,A');      % LMI #2: P'*A'
            lmiterm([-2 2 1 -P],At',-1);    % LMI #2: -At'*P'
            lmiterm([-2 2 2 0],I);          % LMI #2: I
            lmisys=getlmis;

            %---Objective Function Definition---%
            n=decnbr(lmisys) ;
            c=zeros(n,1) ;
            for j=1:n
                %Ei=defcx(lmisys,j,e) ;
                %c(j)=Ei ;
                Si=defcx(lmisys,j,S) ;
                c(j)=trace(Si) ;
            end ;
            %---Lmi Solution---%
            t0=cputime ;
            [copt,popt]=mincx(lmisys,c,options) ;
            %Gama=sqrt(dec2mat(lmisys,popt,e))
            Popt=dec2mat(lmisys,popt,P);
        end
        %%
        n=5;
        P=Popt;
        At=inv(P)*A*P;
        Bt=inv(P)*B;
        Ct=C*P;
        Dt=0;
        Br=Bt(1:r,1);
        Bo=Bt(1+r:n,1);
        Cr=Ct(1,1:r);
        Co=Ct(1,1+r:n);
        sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
        [y1,t1,x1] = step(sys2,t);
        E(k)=norm(y-y1)
```

```
        if E(k)-E(k-1)>=0
            break
        else
            Acnew=Ac;
            %% New Value for Ac
            syms ts
            syms ac1 ac2 ac4 ac5 ac3 ac6
            Acn=[ac1 ac4; ac2 ac5; ac3 ac6];
            % I=-Acn*inv(Ao)*K7*inv(P)*B*u+K6*inv(P)*B*u;
            % xrs=expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-inv(Ar)*I;
            % xs=expm(A*ts)*(x0+inv(A)*B)-inv(A)*B;
            % f=sqrt(int((C*xs-
C*P*K6'*xrs+C*P*K7'*inv(Ao)*K7*inv(P)*B*u)^2,ts,0,tf));
            I=eval(eval(-Acn*inv(Ao)*Bo*u+Br*u));
            xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-
inv(Ar)*I));
            xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
            f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2))
            f=eval(eval(sqrt(int(f1,0,tf))))

            gf(1,1)=subs(eval(eval(diff(f,ac1))),...
                {ac1 ac2 ac3 ac4 ac5 ac6},{Ac(1) Ac(2) Ac(3) Ac(4) Ac(5)
Ac(6)});
            gf(2,1)=subs(eval(eval(diff(f,ac2))),...
                {ac1 ac2 ac3 ac4 ac5 ac6},{Ac(1) Ac(2) Ac(3) Ac(4) Ac(5)
Ac(6)});
            gf(3,1)=subs(eval(eval(diff(f,ac3))),...
                {ac1 ac2 ac3 ac4 ac5 ac6},{Ac(1) Ac(2) Ac(3) Ac(4) Ac(5)
Ac(6)});
            gf(1,2)=subs(eval(eval(diff(f,ac4))),...
                {ac1 ac2 ac3 ac4 ac5 ac6},{Ac(1) Ac(2) Ac(3) Ac(4) Ac(5)
Ac(6)});
            gf(2,2)=subs(eval(eval(diff(f,ac5))),...
                {ac1 ac2 ac3 ac4 ac5 ac6},{Ac(1) Ac(2) Ac(3) Ac(4) Ac(5)
Ac(6)});
            gf(3,2)=subs(eval(eval(diff(f,ac6))),...
                {ac1 ac2 ac3 ac4 ac5 ac6},{Ac(1) Ac(2) Ac(3) Ac(4) Ac(5)
Ac(6)});
%             Ac=Ac-.1*gf/norm(gf(:))
            Ac2=Ac-a0*gf/norm(gf(:));
            I=eval(eval(-Ac2*inv(Ao)*Bo*u+Br*u));
            xrs=eval(eval(expm(Ar*ts)*(K6*inv(P)*x0+inv(Ar)*I)-
inv(Ar)*I));
            xs=eval(eval(expm(A*ts)*(x0+inv(A)*B)-inv(A)*B));
            f1=eval(eval((C*xs-Cr*xrs+Co*inv(Ao)*Bo*u)^2));
            f=eval(eval(sqrt(int(f1,0,tf))));
            alpha=solve(eval(diff(f,a0)),a0);
            Ac=eval(Ac-alpha(1)*gf/norm(gf(:)))
        end
end
Ac=Acnew
P=Pnew;
At=inv(P)*A*P;
Bt=inv(P)*B;
Ct=C*P;
Dt=0;
Br=Bt(1:r,1);
```

```
Bo=Bt(1+r:n,1);
Cr=Ct(1,1:r);
Co=Ct(1,1+r:n);
sys2=ss(Ar,Br-Ac*inv(Ao)*Bo,Cr,-Co*inv(Ao)*Bo); %Solve xr
[y1,t1,x1] = step(sys2,t);
plot(t,y,'r',t1,y1)
hold on

%% Truncation
t=0:0.1:tf;
R1=[1 4 -1; 1 3 -2; 2 2 -1; 2 -1 2; 1 1 -2];
J=[-1 2 -3 4 5; 1 -2 3 -4 6; 1 -2 3 4 7; 1 2 3 4 -8; -1 2 3 -4 9];
T1=J'*R1*inv(R1'*J'*R1);
Art=R1'*A*T1;
Brt=R1'*B;
Crt=C*T1;
Drt=D;
sys=ss(A,B,C,D);
[y,t,x] = step(sys,t);  %solve x
sys2=ss(Art,Brt,Crt,Drt);
[y2,t2,x2] = step(sys2,t);  %solve x
plot(t2,y2,'c')
PP=R1*T1';
grid
hold on
plot(t,y)
```

التخفيض الامثل لرتبة الانظمة الديناميكية باستخدام طريقة مصفوفة التخفيض

إعداد

محمد عبدالله القرعان

المشرف

الدكتور موسى عبدالله

ملخـــــص

في هذا البحث تستخدم طريقة تخفيض الانظمة عن طريق المصفوفة القابلة للتخفيض لتزويدنا بطرق منظمة للحصول على انظمة ميكانيكية مخفضة. ونستطيع من خلال تلك الطريقة الحفاظ على ال (eigenvalues) ذات الاهمية العالية بالنسبة للنظام الاصلي, مما يجعل عملية تخفيض الرتب للانظمة الميكانيكية اكثر قبولا بشكل فيزيائي.

استخدمت طريقة ال (Steepest Descent) للحصول على افضل مكونات للمصفوفة التى ظهرت من خلال طريقة التخفيض باستخدام طريقة المصفوفة القابلة للتخفيض التى تعطى حلا امثلا لعملية التخفيض. الطريقة المقترحة تم اختبارها عن طريق تمثيلها ومن ثم مقارنتها مع طرق تخفيض اخرى معتمدة.